SIEMENS

SCE Training Curriculum

Siemens Automation Cooperates with Education (SCE) | 09/2015

PA Module P01-07

SIMATIC PCS 7 –
Importing Plant Design Data

Cooperates with Education

Automation

SIEMENS

Matching SCE Trainer Packages for these curriculum

- **SIMATIC PCS 7 Software block of 3 packages**
  Order No. 6ES7650-0XX18-0YS5
- **SIMATIC PCS 7 Software block of 6 packages**
  Order No. 6ES7650-0XX18-2YS5
- **SIMATIC PCS 7 Software Upgrade block of 3 packages**
  Order No. 6ES7650-0XX18-0YE5 (V8.0 → V8.1) or 6ES7650-0XX08-0YE5 (V7.1 → V8.0)
- **SIMATIC PCS 7 Hardware Set including RTX Box**
  Order No. 6ES7654-0UE13-0XS0

Please note that these trainer packages may be replaced with subsequent packages.
An overview of the available SCE packages is provided at: siemens.com/sce/tp

Continuing education

For regional Siemens SCE continuing education, contact your regional SCE contact partner.
siemens.com/sce/contact

Additional information relating to SIMATIC PCS 7 and SIMIT

In particular, Getting Started, videos, tutorials, manuals wband programming guide.
siemens.com/sce/pcs7

Additional information relating to SCE

siemens.com/sce

Note on Usage

The training curriculum for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)“ specifically for training purposes at public educational and R&D facilities. Siemens AG is not liable for the contents.

This document may only be used for initial training on Siemens products/systems. This means it may be copied entirely or partially and handed to trainees for use within the scope of their training. Passing on or copying this document and communicating its contents is permitted within public training and continuing education facilities for training purposes.

Exceptions require written permission by Siemens AG. Contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Violators are subject to damages. All rights including translation rights are reserved, particularly in the event a patent is granted or a utility model or design is registered.

Usage for industrial customer courses is explicitly not permitted. We do not agree to the commercial utilization of these documents.

We would like to thank the Technical University Dresden, particularly Prof. Dr. Leon Urbas and Annett Krause, MS, as well as the Michael Dziallas Engineering Corporation and those who provided support in preparing this SCE training document.

IMPORTING PLANT DESIGN DATA

TRAINING OBJECTIVE

The students learn to identify recurrent structures and to design templates. They know the difference between a process tag type and a model. They will be able to create and implement both. This allows the students to implement many similar process tag types or units in ***PCS 7***. They become familiar with the process object view and are able to use it to represent parameters system-wide, and change them if needed.

THEORY IN BRIEF

In process engineering plants, objects and structures recur again and again that behave in the same way, that are equally integrated in control engineering, and that are to be visualized in the same way.

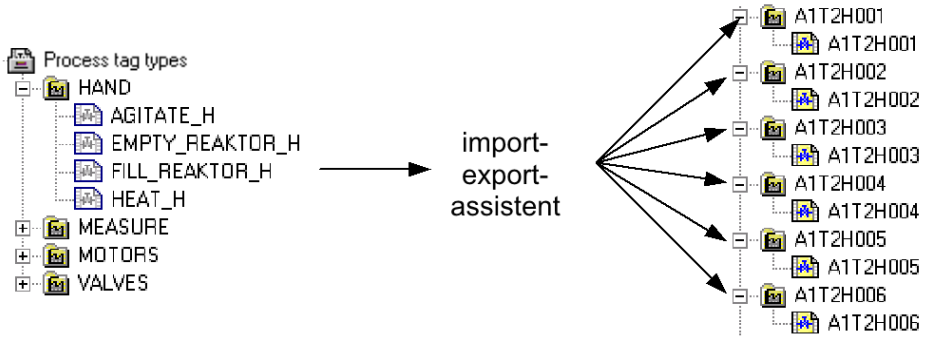


Figure 1: From process tag types to replicas

Such an object can be stored in the project library as ***process tag type***. A process tag type is a single CFC. As shown in Figure 1, a large number of process tags can be generated in one operation as a copy of one process tag type, using the import/export wizard. This process is controlled by an import file. Then, the process tags can be manually adapted and connected correspondingly to specific automation tasks.

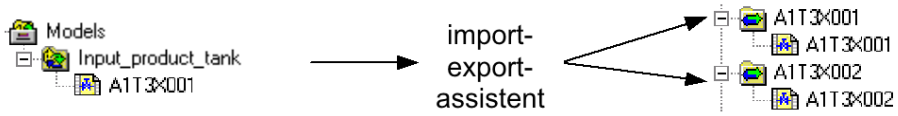


Figure 2: From models to replicas

With ***models*** we define more complex functions than with process tag types (up to complete units). A model consists of hierarchy folders containing CFC/SFCs, pictures, reports and supplementary documents. The entire structure can be stored in the project library as a re-useable template. Based on an import file, a large number of replicas can be generated as copy from a model in one operation using the import/export wizard (refer to

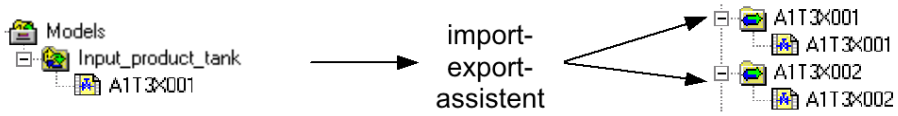


Figure 2). Then, the replicas are adapted to the specific requirements of the respective automation task.

The ***PCS 7*** libraries contain extensive ***templates***. If a template is to be used multiple times, it is copied from the ***PCS 7*** library to the project library, adapted if needed and copied by means of the import/export wizard based on an import file.

THEORY

When designing an automation system with ***PCS 7***, we can resort to general design principles for complex systems that have proven themselves again and again [1]. The most important principles are:

– The principle of hierarchical arrangement

– The principle of modularization

– The principle of reuse

The principle of hierarchical arrangement was used previously when we structured the plant in the chapter 'Plant Hierarchy'. Through structuring into subsystems that can be processed largely independent of each other, a design problem that initially seemed unmanageable is broken down into sub-tasks that are manageable and can be planned.

The principle of modularization implies that a system to be designed is set up with constituent parts (here: blocks, CFCs, SFCs) that have the following characteristics:

– The scope is manageable and can be followed easily

– Largely autonomous functions that can be checked

– As few relationships to other constituent parts as possible

– Defined interfaces to other constituent parts

This results in two rivaling complexity aspects when an automation solution is broken down into its parts:

– Low inner complexity of the parts: The more parts, the smaller and more manageable the individual parts.

– High exterior complexity of the parts: The more parts, the higher the number of connections between the parts.

Hierarchical structuring and modularization depend on each another. While hierarchical structuring is determined more by the process engineering system, modularization is dominated by process control engineering implementation. Based on the countercurrent complexity aspects mentioned above and the high dependency on actual process engineering and automation engineering tasks to be solved, early coordination of both is of advantage.

Through the plant hierarchy, ***PCS 7*** supports the principle of hierarchical structuring. The principle of modularization and reuse is realized in ***PCS 7*** in importing plant design data.

In larger projects or in the case of recurring similar projects, often a large number of identical or at least very similar objects and structures can be observed. To save time and outlay for the configuration, it is advisable to plan the specific search for suitable, recurring objects and structures in the concept phase and the design phase of an automation project. After such objects and structures are identified, first generic solutions are implemented and tested that subsequently can be used for a variety of identical or similar objects and structures. The additional effort that the preparation of the generic solution (here also called types and templates) entails should lead to considerable time and cost savings over the overall duration of the project because of the following factors:

– A type can be implemented multiple times, which means it has several replicas.

– By using a type in several replicas, several tests are performed at the same time.

– If errors should occur or changes are necessary, the generic solution only has to be adapted and all replicas updated.

Moreover, objects and structures that are available from earlier projects and libraries can be reused. Their advantage is that they have been tried and are largely free of errors. By using well-tried parts, the reliability of a new automation solution in general rises.

PROCESS TAG TYPE

The process tag type is used as a generic solution when a project contains many process tags of the same kind [2].

First, a CFC is prepared that contains all internal blocks and their interconnections. All input and output parameters are defined uniquely as parameters or signals. This CFC with all generally valid parameters is used to generate a process tag. In a so-called import file, the process tag specific parameters are specified in which the replicas differ.

During the import, the import/export wizard generates the process tag type replicas in the specified hierarchy folders. If there is no hierarchy, it is set up as well. Each replica is an instance of the process tag type and has its properties.

In ***PCS 7***, the process tags (replicas) generated in this way can be specifically adapted in addition by adding, for example, different interlocking mechanisms. Under certain preconditions, these are not overwritten even if they are re-imported.

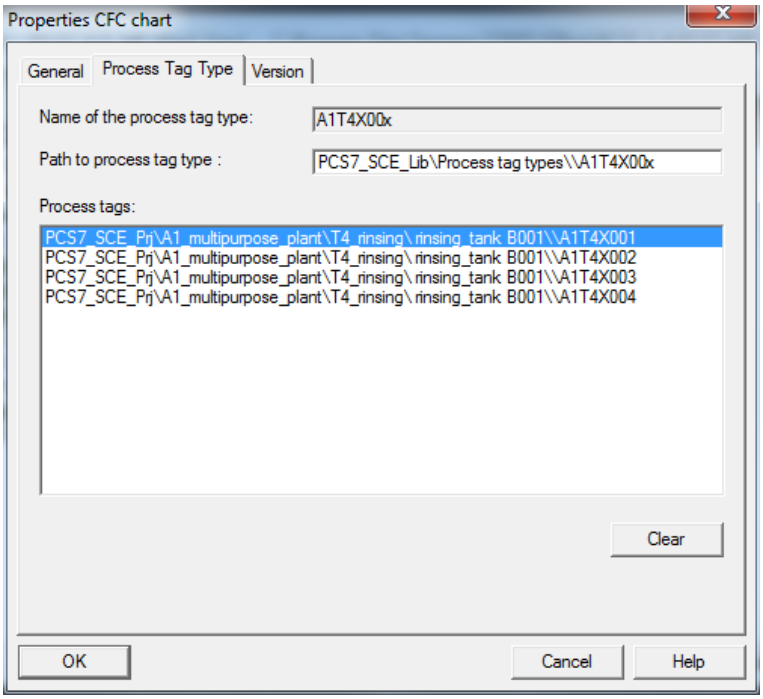


Figure 3: Replica A1T2H003 of FILL_REACTOR_H

The following must not be changed for the process tags that were generated:

– Specific adaptations to the block interconnections that are parameterized by means of the import file. These adaptations are overwritten at the next import with the parameters that are specified in the import file.

– Block name changes.

Regarding process tag types, subsequent changes can be made easily by performing them at the process tag type and the import file. Then, the modified data is transferred to all process tags with another import. The following changes are conceivable:

– Supplementing a parameter and assigning this parameter via the import file

– Clearing all generated process tags of a process tag type (without manual deletion in the plant hierarchy)

– Supplementing an additional block interconnection and parameterizing it through the import file

MODEL

The model is used as a generic solution when structures of the same kind occur in the project.

As a rule, a plant is structured by breaking it down into smaller functional units whose interfaces, performance and logic can be clearly described; for example, a tank with its instrumentation. Instead of implementing these functional units again each time, an inventory of pre-assembled functional units (models) can be set up.

For a model to be used project wide in only one version, all models should be stored centrally in the master data library and adapted prior to generating replicas.

A model consists of hierarchy folders with the following elements:

– CFCs/SFCs

– OS pictures

– OS reports

– Additional documents

After a model was configured and an import file was assigned to it, replicas can be generated by means of an import. The following steps are performed automatically:

Step 1: The hierarchy path in the ‘Hierarchy’ column of the first data row in the import file is read. A check determines whether this path already exists. Further action depends on the check result.

– If the hierarchy folder exists and it is a replica of the model, the parameter settings are used from the import file for the existing replica.

– If the hierarchy folder exists and is suitable as a replica of the model, it is made into a replica of the model with its CFC and parameterized according to the import file.

– If there is no hierarchy folder, it is set up. A replica of the model is generated and parameterized accordingly.

Step 2: The following elements are inserted in the title block of the charts if the columns exist:

– Function designation

– Location designation

– CFC name

– Chart comment

Step 3: Texts and values of the parameter descriptions and the interconnection descriptions (signals) are written to the corresponding block or chart connections of the replicas.

Note: An interconnection is deleted when the signal name (symbol or textual interconnection) consists of the code word '---' (three dashes).

An interconnection remains unchanged when no connection name (symbol or textual interconnection) is specified.

Step 4: The data types of the connections for signals are determined and assigned to the interconnections.

Note: The following applies to interconnections with global addresses: When the option 'Enter signal also in the symbol table' is set, the names are searched for in the symbol table of the model resource.

For ***PCS 7*** it is recommended not to use this option because these entries are made in ***HW Config*** when the hardware is configured.

Note the following rules:

– The symbol name is present in the symbol table:

  The data type has to be the same, the symbol name must exist only once. The data type is parameterized according to the block/chart connection. The absolute address is overwritten and the symbol comment is entered for the symbol (if provided in the import file). Only what has changed is overwritten; existing attributes are retained.

– The symbol name does not exist in the symbol table:

  The interconnection is set up and the data type parameterized according to the connection. The absolute address and the symbol comment are entered for the symbol (if it exists in the import file).

Step 5: For each message, the message text is imported.

Then, steps 1 to 5 are repeated for each row of the import file.

When a hierarchy folder was highlighted that contains several models, the import files are displayed each with the model in the list. If needed, the list can be edited. Then, the import is performed for all models in the list as described above.

PARAMETERS AND SIGNALS

For process tag types and models to be generated successfully, it is important to define all inputs and outputs of the CFC as parameter or as a signal. Only connections that are defined as parameter or as signal can be included in the column of the import file and parameterized.

PROCESS OBJECT VIEW

With the process object view, all data of the basic automation are represented project wide in a control oriented view. Project wide means that the data of all included projects is recorded in a multi-project.

The process object view is structured similar to the plant hierarchy:

– In the left half of the window, the plant hierarchy is represented as a tree structure (hierarchy window). There, identical operating options are provided. In addition, the CFCs, SFCs, pictures, reports and supplementary documents are displayed in the hierarchy window.

– In the right half, a table of the lower level objects with their attributes is displayed (content window). The content window has the tabs shown in Table 1 and provides different views to the project data.

Table 1: Tabs of the process object view

Tab	Usage
General	This tab displays all lower-level ES objects (process tags, CFCs, SFCs, pictures, reports, or additional documents) and their general information for the plant unit currently selected in the tree view.
Blocks	This tab displays the block properties of the blocks of all lower-level CFCs for the plant unit currently selected in the tree view. In this context, SFC instances are also referred to as blocks.
Parameters	This tab displays the I/O points that were explicitly selected for editing in the process object view (S7_edit = 'para') for all the process tags and CFCs displayed in the "General" tab.
Signals	This tab displays the I/O points that were explicitly selected for editing in the process object view (S7_edit = 'signal') for all the process tags and CFCs displayed in the "General" tab.
Messages	This tab displays the corresponding messages for all the process tags, CFCs and SFCs displayed in the "General" tab.
Picture objects	This tab displays any picture interconnections which may exist in ***WinCC*** (if available) for all the process tags and CFCs displayed in the "General" tab.
Archive tags	This tab displays any existing interconnected ***WinCC*** archive tags with their attributes for all the process tags, CFCs and SFCs displayed in the "General" tab. Only those attributes that are relevant for ***PCS 7*** (subset of all attributes defined in Tag Logging).
Hierarchy folder	This tab displays the hierarchy folders of the plant unit selected in the tree view (one line per hierarchy folder).
Equipment properties	This tab displays the equipment properties for the project selected in the tree view. These equipment properties are instances created by the equipment property types configured in the shared declarations (one line per equipment property. In case of a type change, that attributes are applied at the instance.
Shared Declarations	This tab shows the attributes of the enumerations, units and equipment properties included the multiproject.

LITERATURE

[1] Lauber, R. und Göhner, P. (1999): Prozessautomatisierung 2. Springer Verlag

[2] Online Help for PCS 7. Siemens

STEP BY STEP INSTRUCTIONS

TASK

PCS 7 is a software that provides users with many tools for programming large plants and duplicating program parts.

In this task, charts and hierarchy structures are created as library objects. They can then be used multiple times. The import/export wizard and the process object view are used to help with the task.

The chart for valve ‘A1T2X001’ is used here as process tag template. All additional valve inlets for the reactors are created using this process tag.

For the model, use educt tank B001 and create from it all additional educt tanks.

TRAINING OBJECTIVE

In this chapter, the student learns the following:

– Importing plant design data using the import/export wizard

– Familiarization with the process object view

– Copying charts by generating process tags

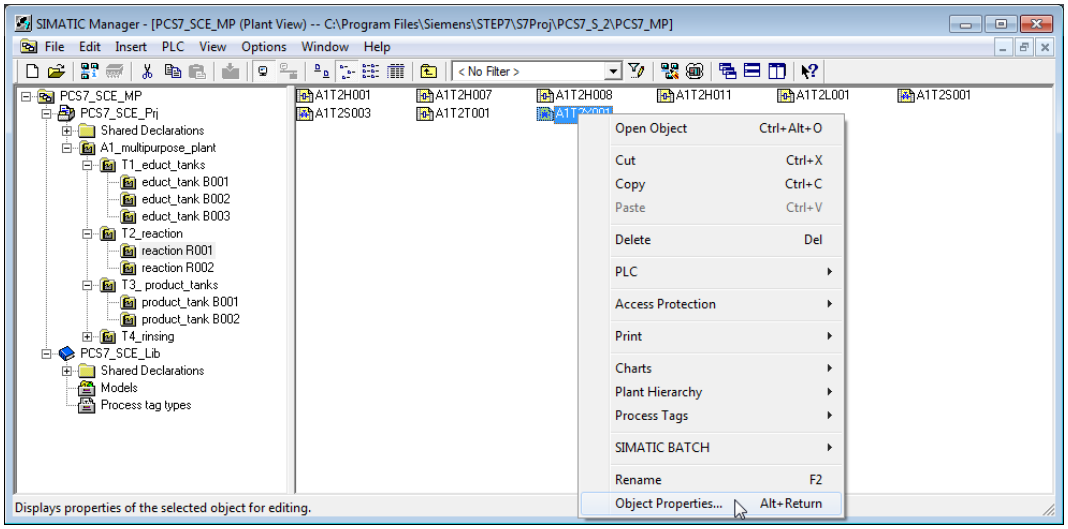
– Copying folder structures by creating models

These instructions are based on the project ‘PCS7_SCE_0106_Ueb_R1503_en.zip‘.

PROGRAMMING

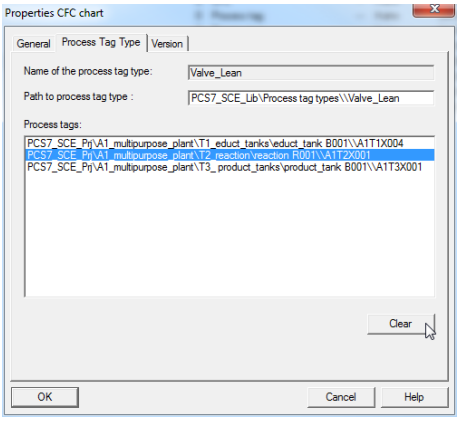
1. To duplicate a chart that is already created and tested, a process tag is generated from it. In this example, we are using valve ‘A1T2X001’. Because this chart is already associated with the process tag type ‘Valve_Lean‘, we first have to cancel the connection in the object properties.

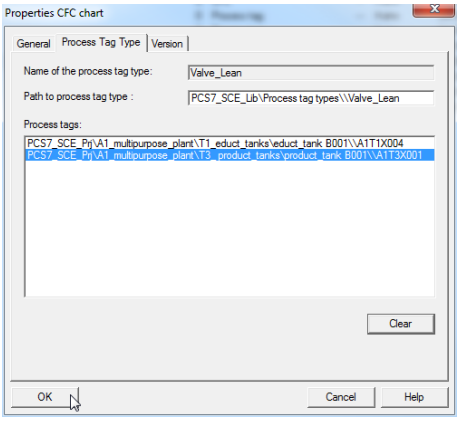
   (→ A1T2X001 → Object Properties)



2. In the ‘Process tag type‘ tab, highlight the row with the valve and then click ‘Cancel‘. The valve is removed from the list.

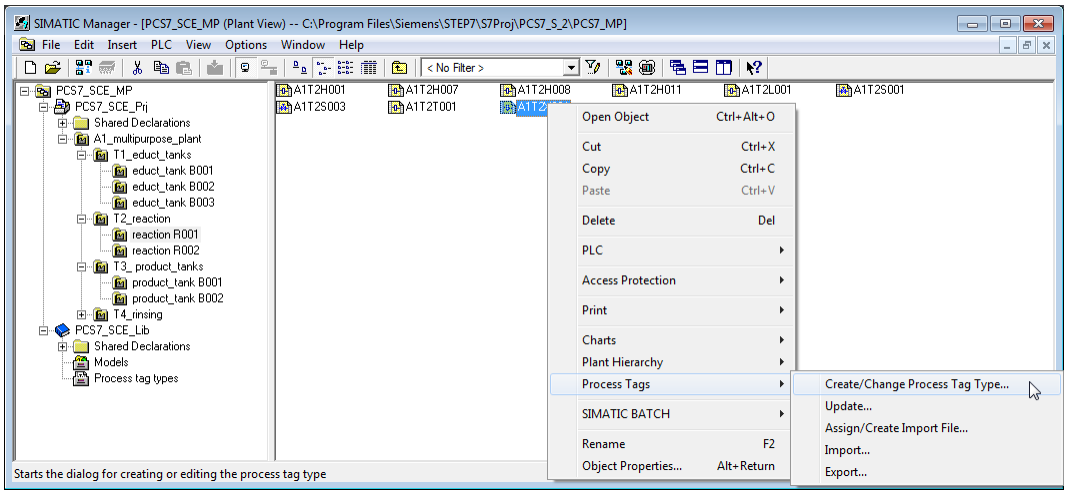
   (→ Process tag type → A1T2X001 → ‘Cancel‘→ ‘OK‘)



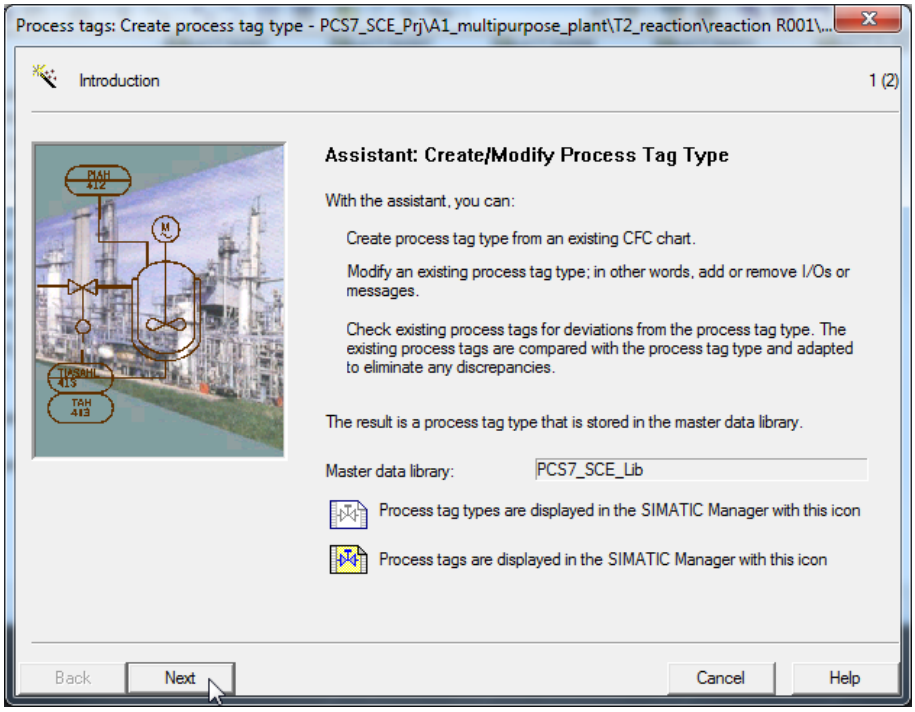


3. Now we can generate a process tag type from ‘A1T2X001‘ by clicking on ‘Process tags’ in the shortcut menu and then on ‘Create/change Process Tag Type…‘.

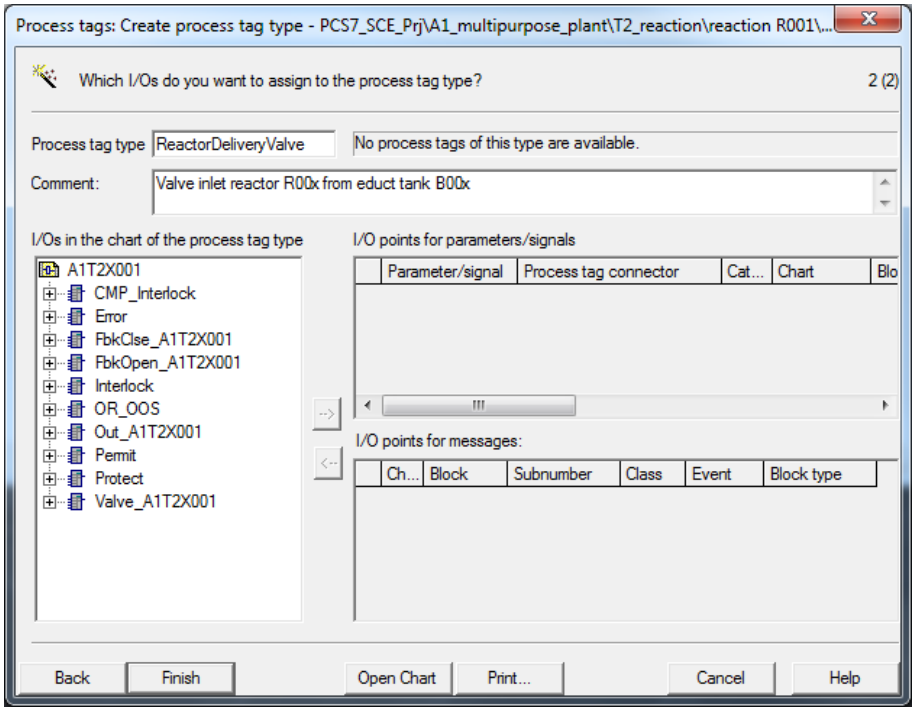
   (→ A1T2X001 → Process Tags → Create/Change Process Tag Types…)



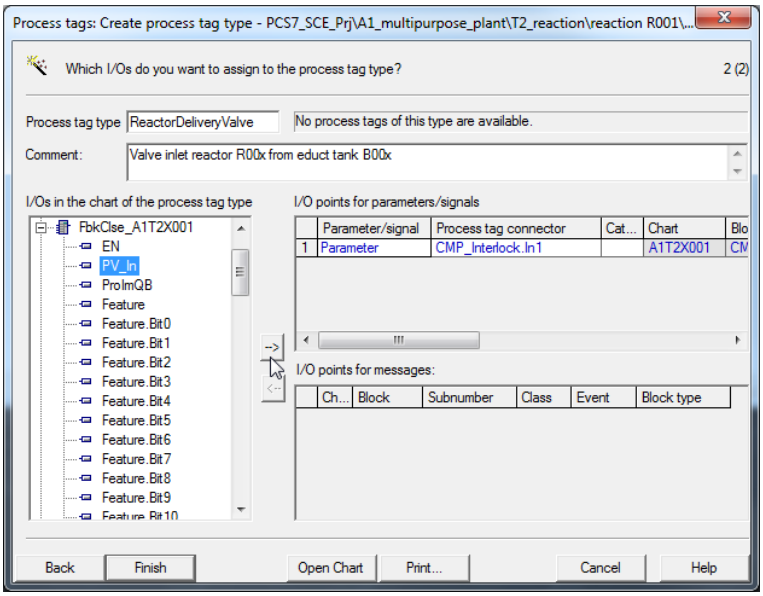
4. The dialog Create/Modify Process Tag Type opens. (→ Next)



5. First, the name of the process tag type is generalized to ‘ReactorDeliveryValve‘ and the comment to ‘Valve inlet reactor R00x from educt tank B00x‘.



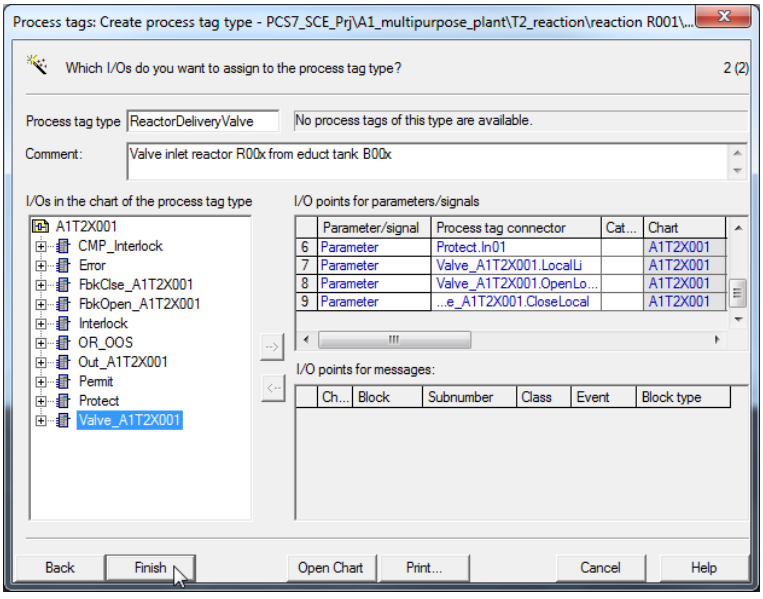
6. Next, the parameters and signals that have to be changed between the individual replicas of the process tag type have to be selected on the left side of the window. (FbkClse_A1T2X001 → PV_In → -->)



Note: With "Open Chart" the associated CFC is displayed to get a better overview. (→ Open Chart)

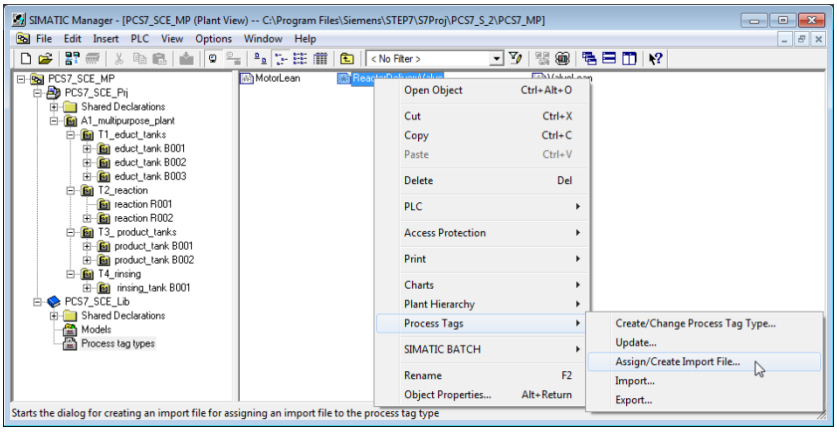
7. Now, add all signals and parameters that represent I/O points of the CFC. Signals are input and output signals, and parameters are interconnections between charts. The signals and parameters shown here have to be added for the valve inlets of the reactors. Then the process tag can be finished. (→ Finish)

	Parameter/signal	Process tag connector	Cat...	Chart	Block	I/O name	I/O comment	Data type	I/O	Block type
1	Parameter	CMP_Interlock.In1		A1T2X001	CMP_Interlock	In1	Analogue Value 1	STRUCT	IN	CompAn02
2	Signal	FbkClse_A1T2X001.PV_In		A1T2X001	FbkClse_A1T2X001	PV_In	Input value	BOOL	IN	Pcs7DiIn
3	Signal	FbkOpen_A1T2X001.PV_In		A1T2X001	FbkOpen_A1T2X001	PV_In	Input value	BOOL	IN	Pcs7DiIn
4	Signal	Out_A1T2X001.PV_Out		A1T2X001	Out_A1T2X001	PV_Out	Output value	BOOL	OUT	Pcs7DiOu
5	Parameter	Permit.In01		A1T2X001	Permit	In01	Input 01	STRUCT	IN	Intlk02
6	Parameter	Protect.In01		A1T2X001	Protect	In01	Input 01	STRUCT	IN	Intlk02
7	Parameter	Valve_A1T2X001.LocalLi		A1T2X001	Valve_A1T2X001	LocalLi	1=Local Mode: Local operation by field signal	STRUCT	IN	VlvL
8	Parameter	Valve_A1T2X001.OpenLo...		A1T2X001	Valve_A1T2X001	OpenLocal	1=Open Local:Field Open Signal	STRUCT	IN	VlvL
9	Parameter	...e_A1T2X001.CloseLocal		A1T2X001	Valve_A1T2X001	CloseLocal	1=Close Local: Field Close Signal	STRUCT	IN	VlvL

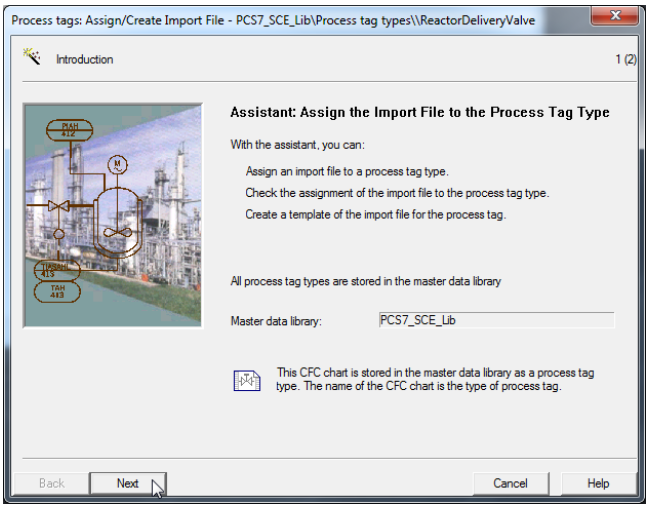


8. After Finish, the process tag type is located in the plant view in the project library under “Project Tag Types“. We now have to create an import file for the project tag type we have just created.

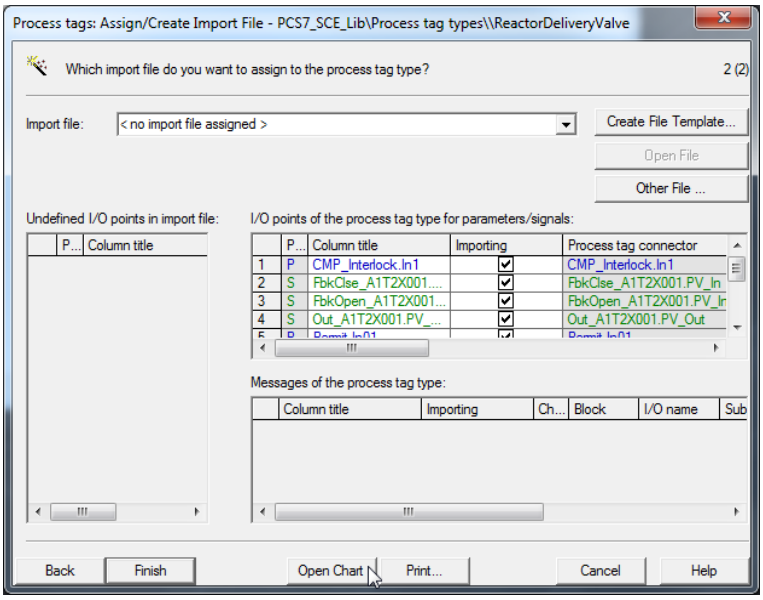
   (→ Project Tag Types → Project Tags→ Assign/Create Import File)



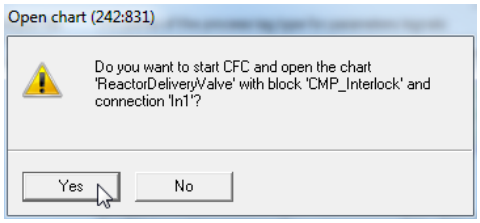
9. The first dialog is confirmed with “Next“. (→ Next)



10. First you have to open the chart. (→ Open Chart)



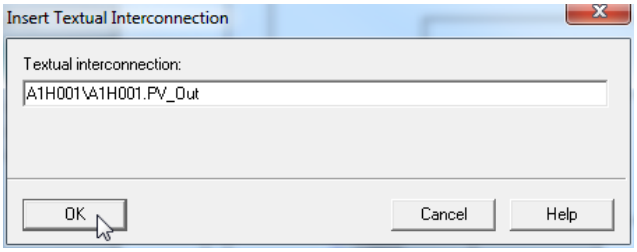
11. Confirm the message that follows. (→ Yes)

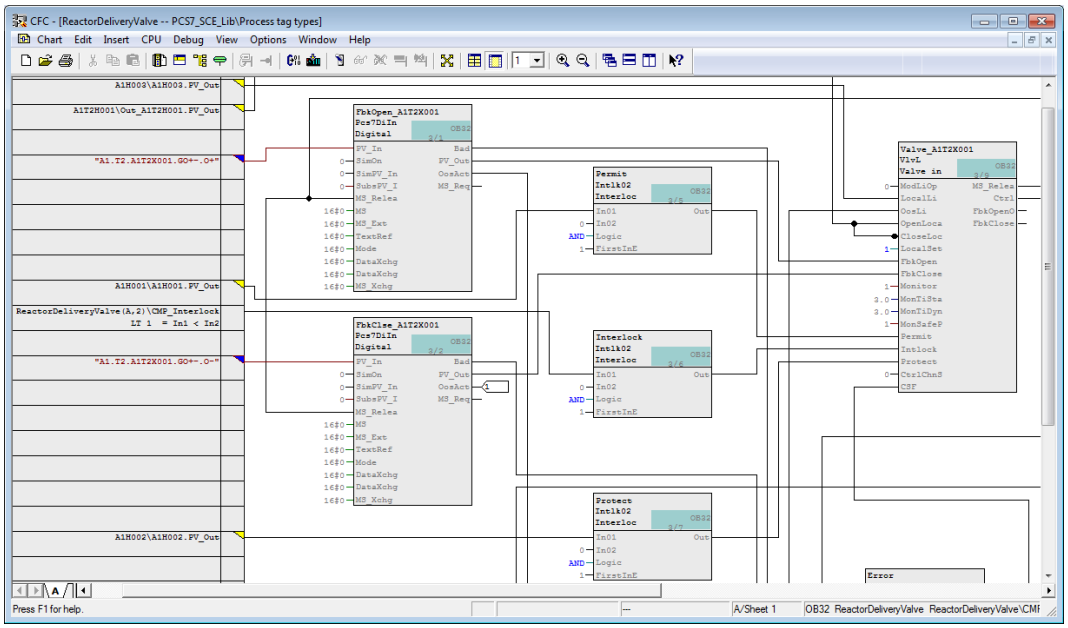


12. You can see that all cross-chart connections are set up as textual interconnections, and all input and output signals with their symbolic names. The chart can now be closed again. (→ Close)

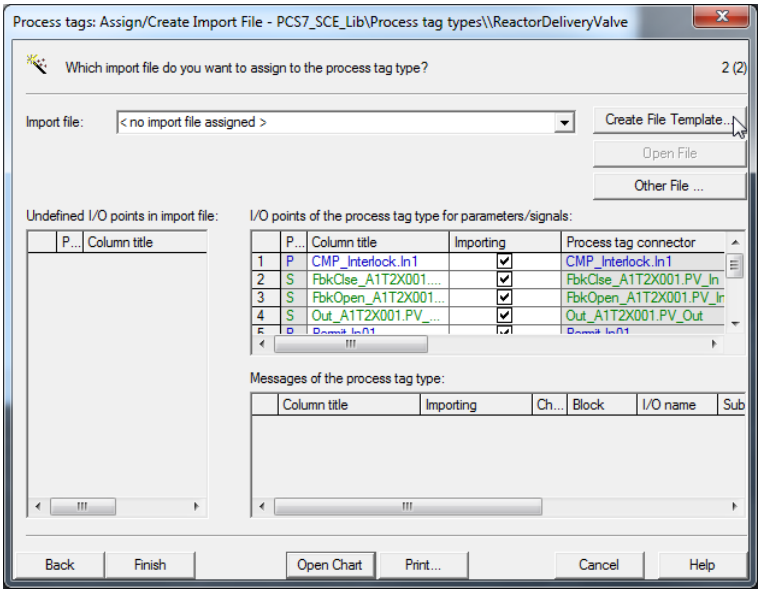
Note: The textual interconnection A1H001\A1H001.PV_Out is structured as follows:

A1H001	Name of CFCs
\	Separator
A1H001	Name of block in the CFC
.	Separator
PV_Out	I/O of the block that is to be connected

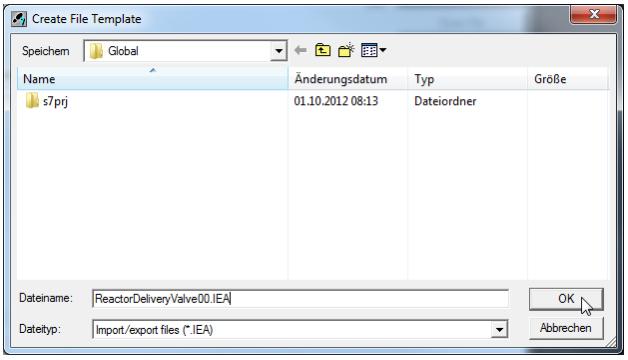




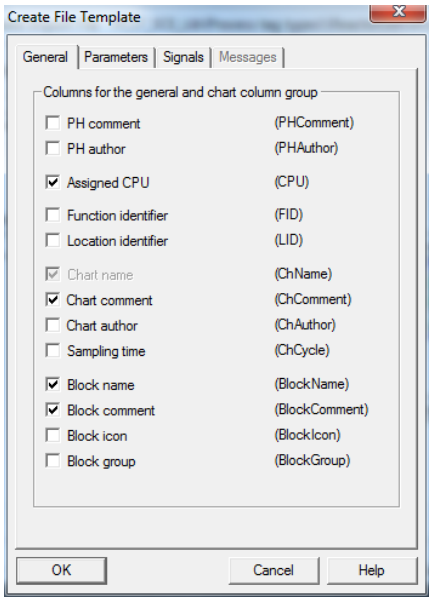
13. Next, create a new file template. (→ Create File Template…)



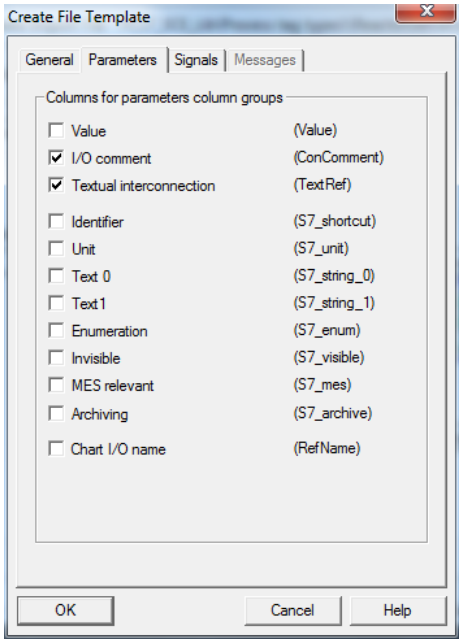
14. To the import file we assign the name ReactorDeliveryValve00.IEA and select a memory location. (→ OK)

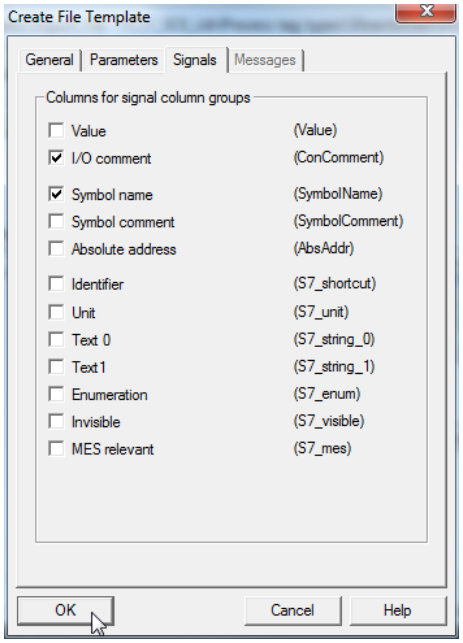


15. Next we select the general columns that are to be displayed in the import file. (→ General → Assigned CPU → Chart comment → Block name → Block comment)

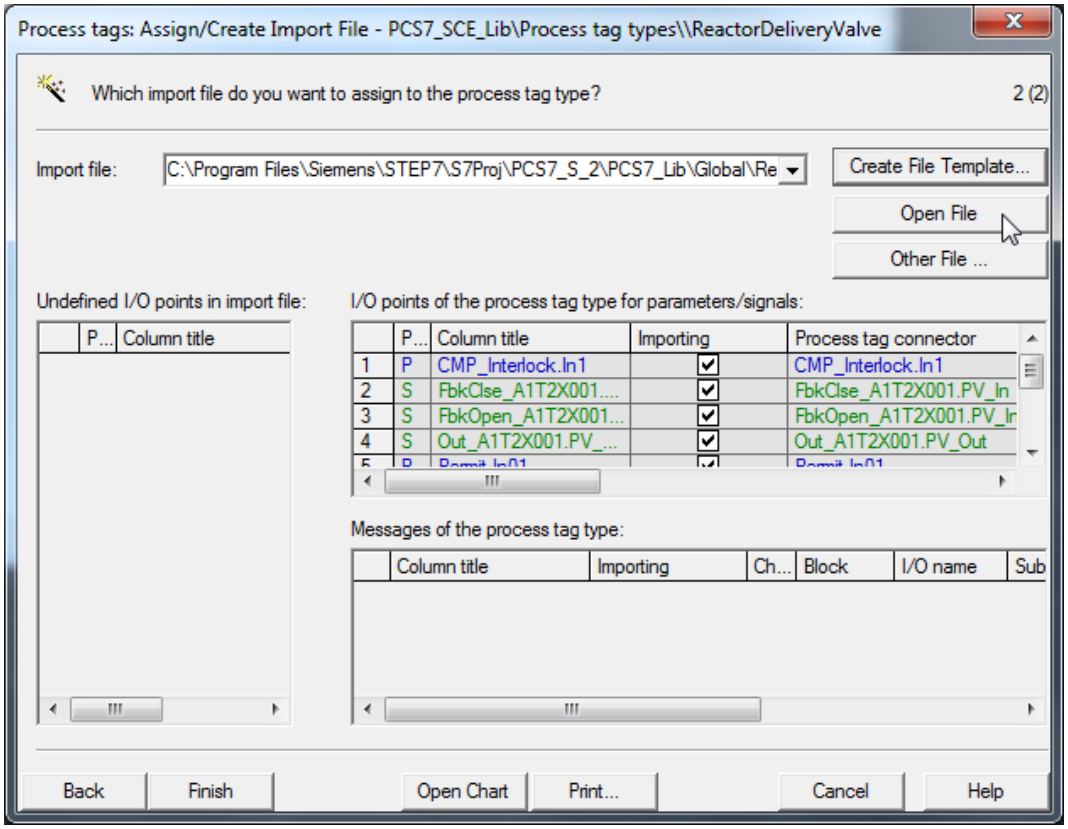


16. Then we select the columns that are to be displayed for the parameters and the signals in the import file. (→ Parameters → I/O comment→ Textual interconnection → Signals → I/O comment→ Symbol name → OK)

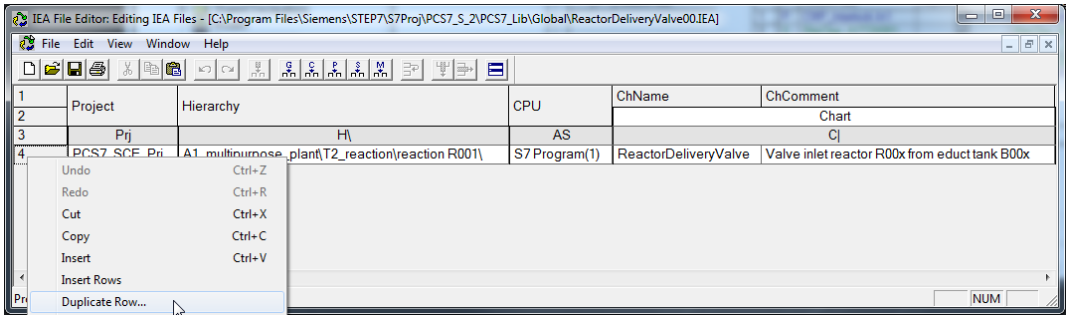




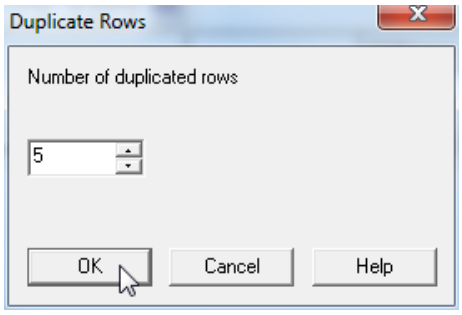
17. The import file created in this way is then opened. (→ Open File…)

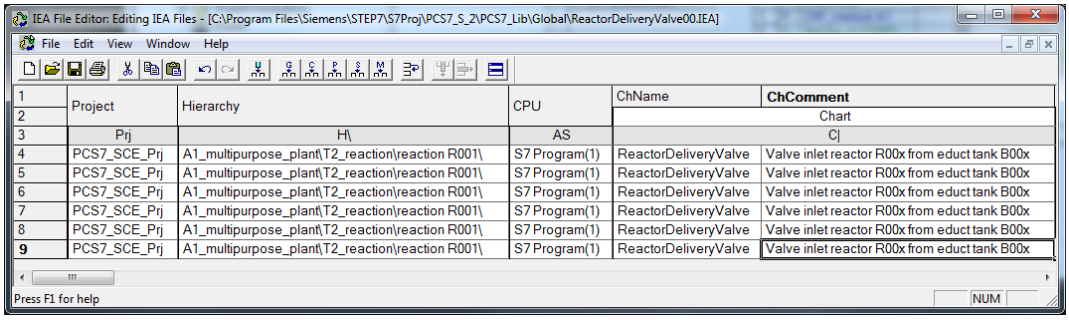


18. Now, duplicate the first row by selecting, after a right click on the first row, the option “Duplicate row…”. (→ Duplicate Row…)

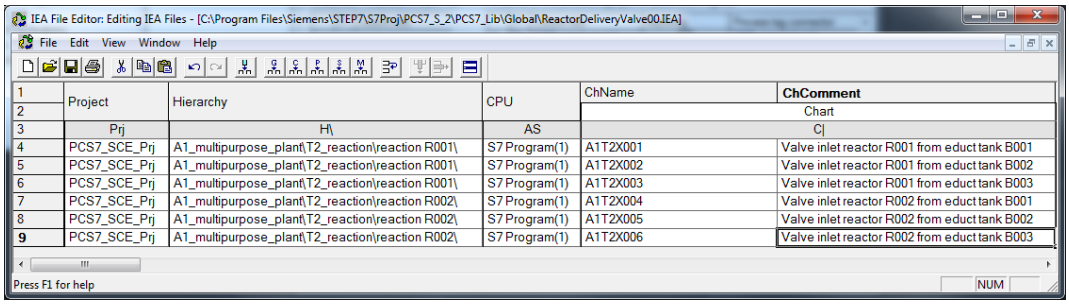


19. In the window that now opens, enter the number of rows. In this case there are 5, because a total of 6 valve inlets exist for the reactors that are to be edited/created using this process tag type. (→ 5 → OK)

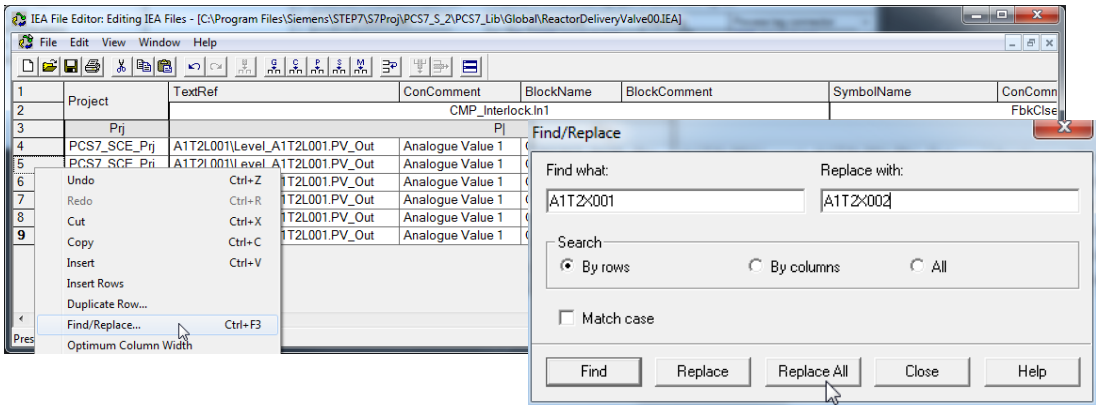




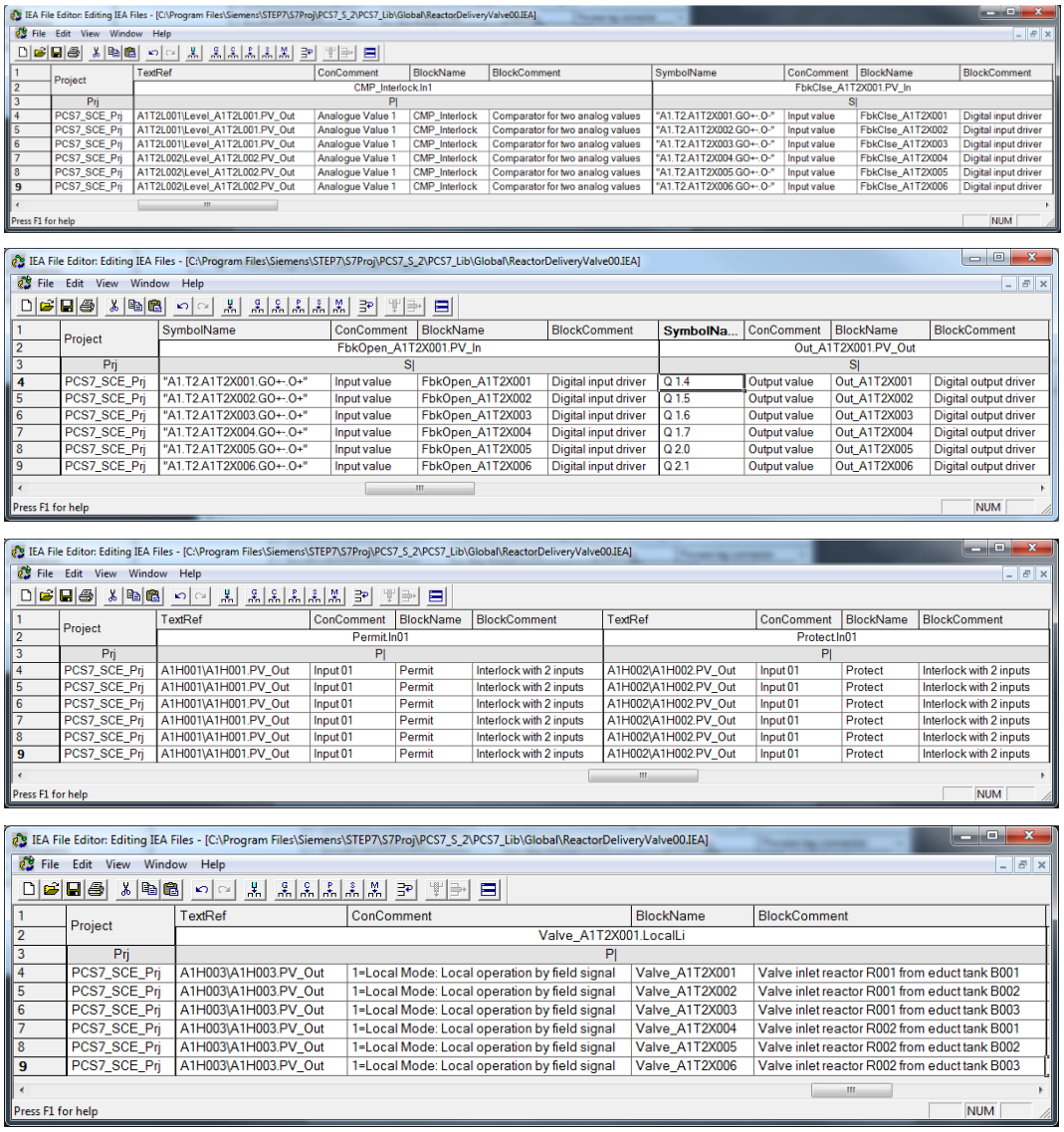
20. In the duplicated rows, we now enter the specific properties for each valve. Start with the hierarchy, the ChName and ChComment.



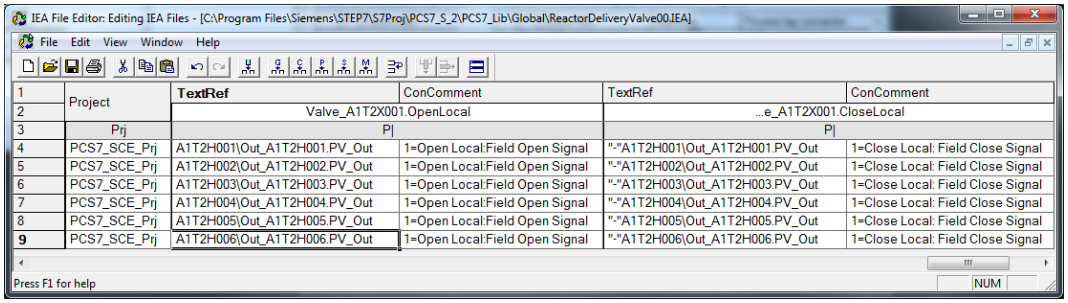
21. Next, we have to set the correct parameters and signals for each row. This can be speeded up by using the row by row Find/Replace. In row 2, for example, we can replace ‘A1T2X001‘ with ‘A1T2X002‘.



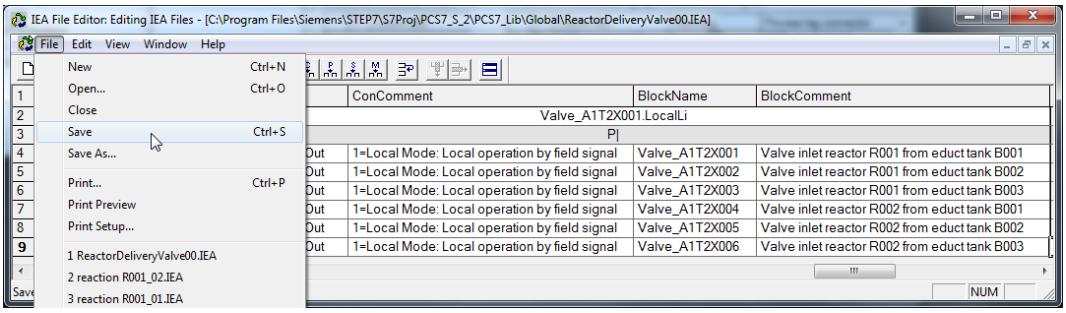
22. Now, edit the rows of the file as shown below. The input signals (SymbolName column) should be placed in quotation marks ““; otherwise, they cannot be located. The output signals (SymbolName column) should be set as absolute address, or the CFCs corrected afterwards.



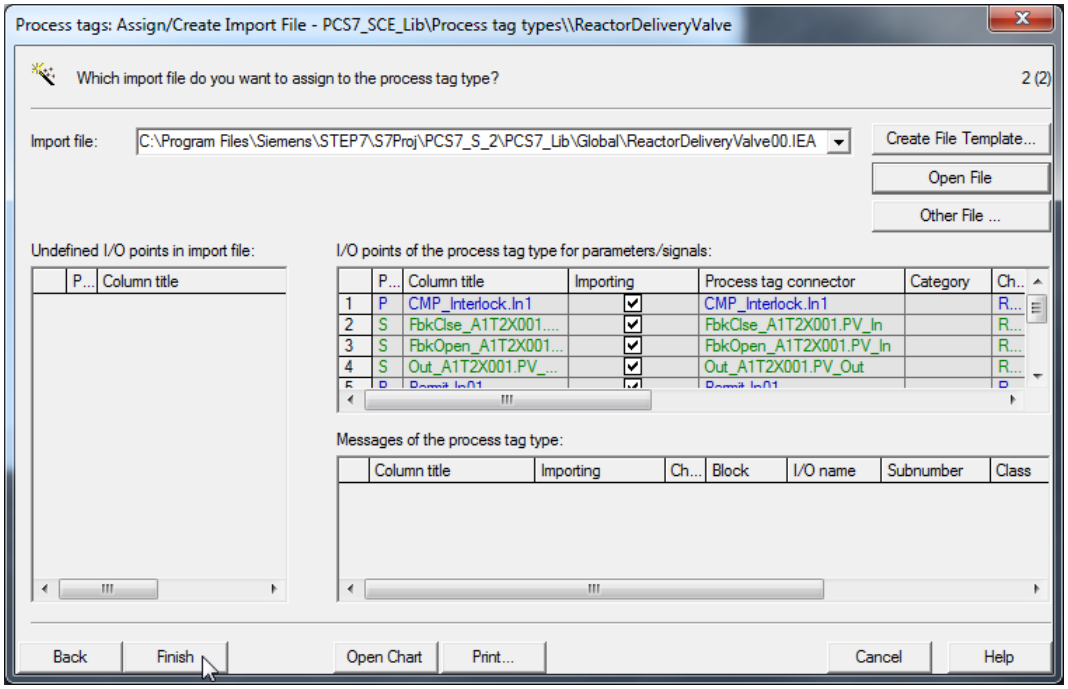
23. Finally, change the parameter for the manual control as shown here. The character "-" in front of the textual interconnection means 'invert'.



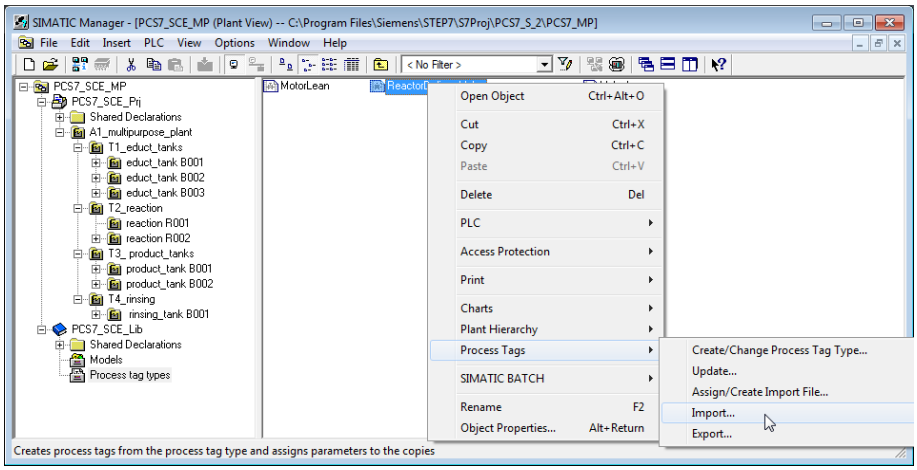
24. After all changes have been made, save the file. (→ File → Save → Close)



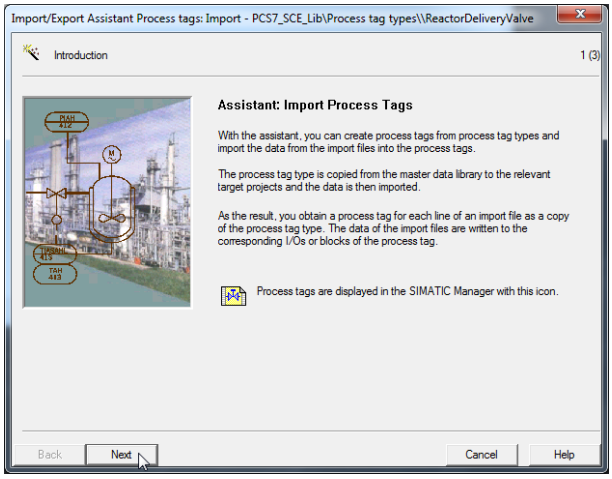
25. Creating and assigning the import file is now finished. (→ Finish)



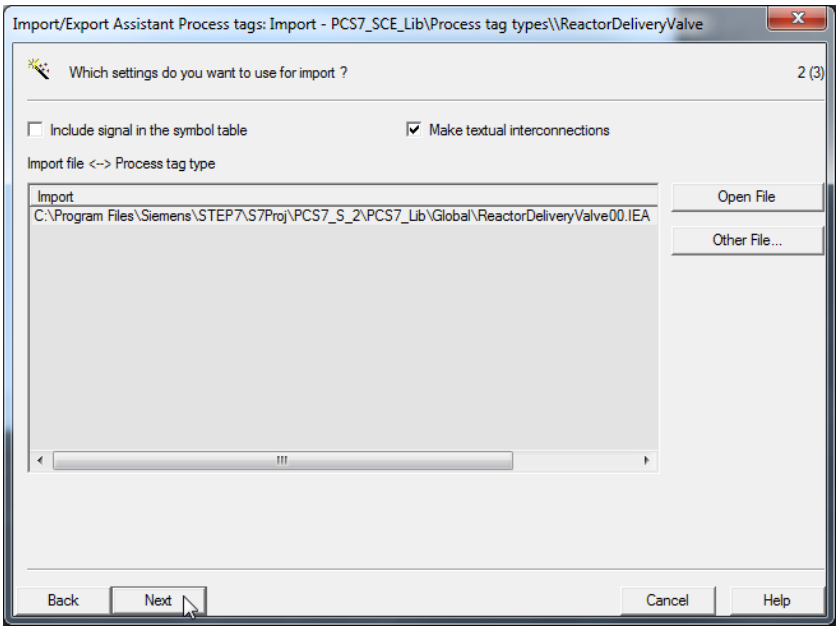
26. We can now start importing the created process tag type. (→ ReactorDeliveryValve → Process Tags → Import…)



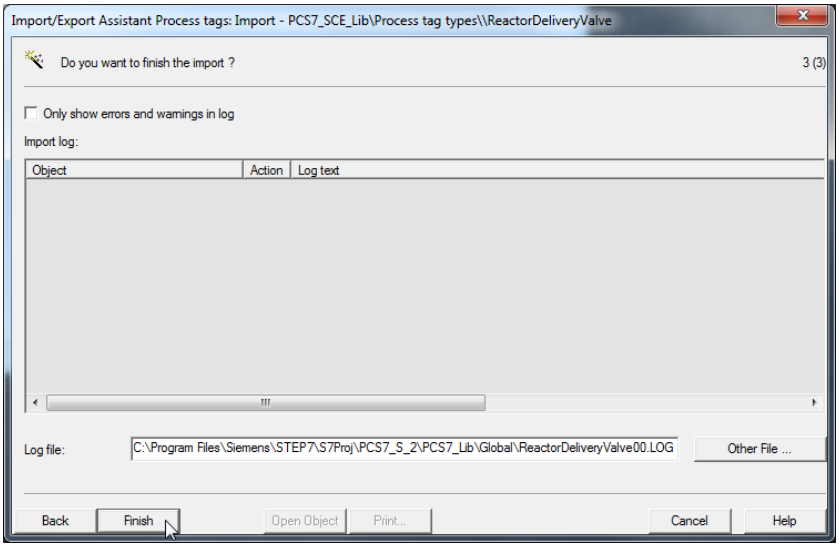
27. The first step of the dialog is confirmed with “Next“. (→ Next)



28. In the next dialog box, we select the option “Make Textual Interconnections“ and then click on “Next“. (→ Make Textual Interconnections → Next)

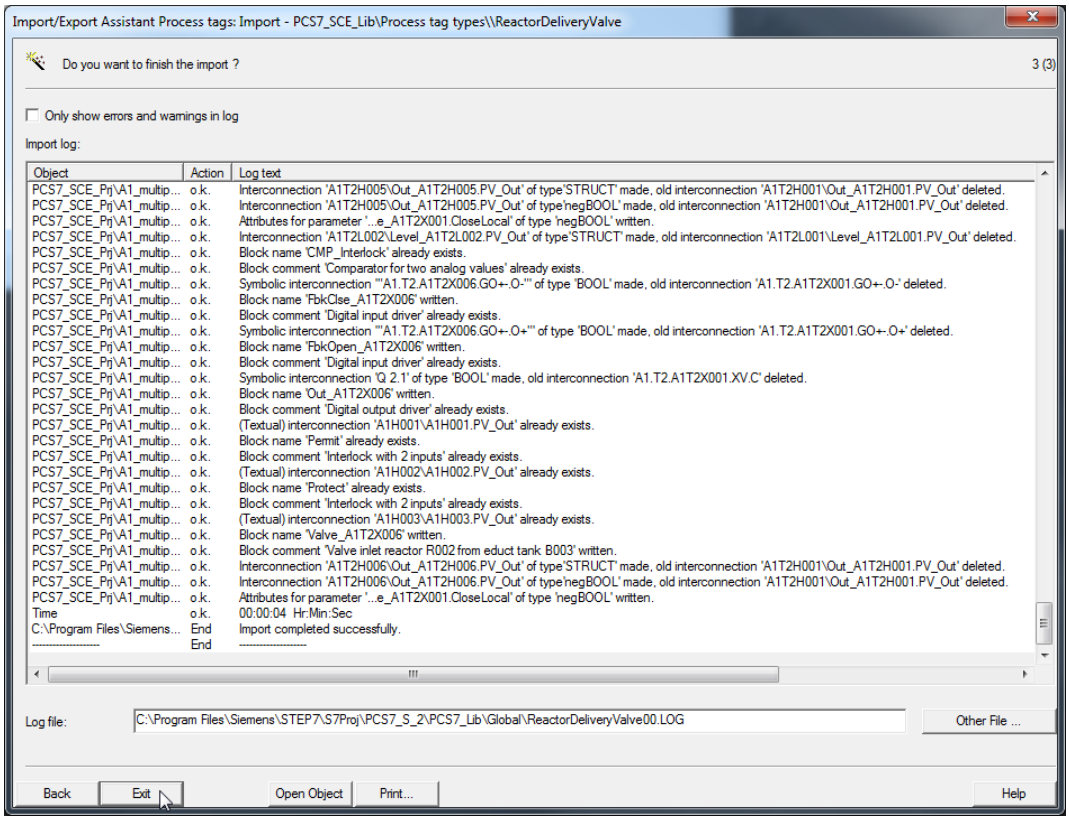


29. We can now start the import by selecting “Finish“. (→ Finish)

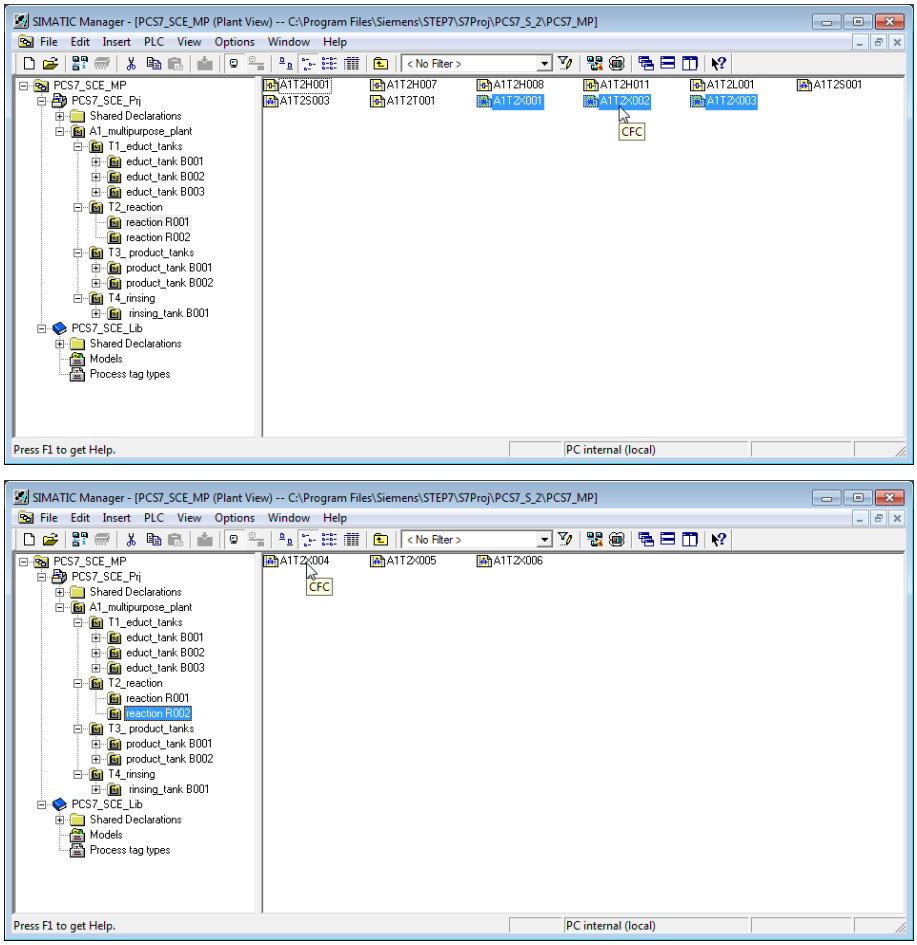


30. After this process is completed, the log is displayed.

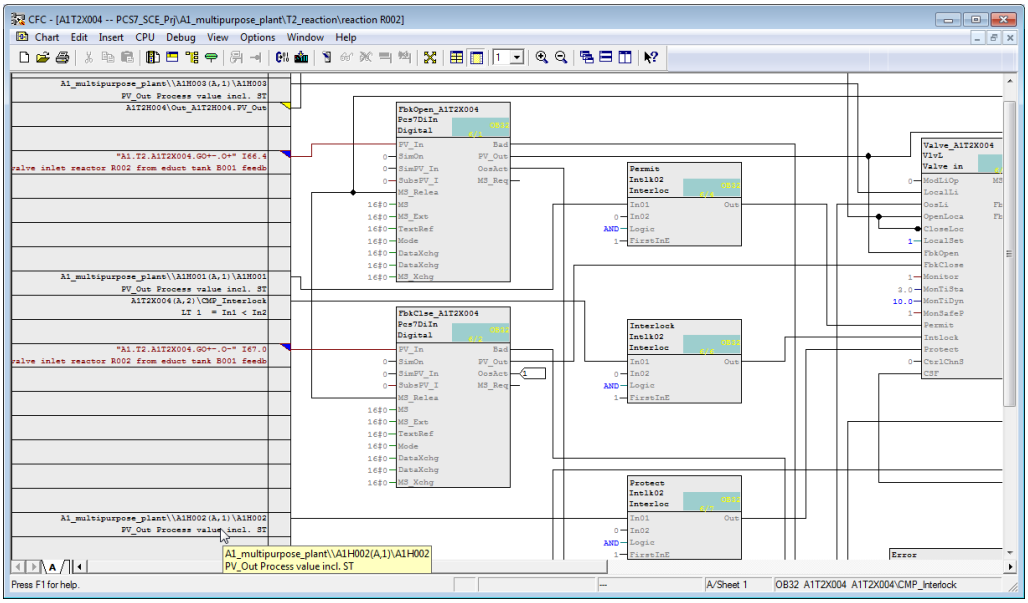
(→ Exit)

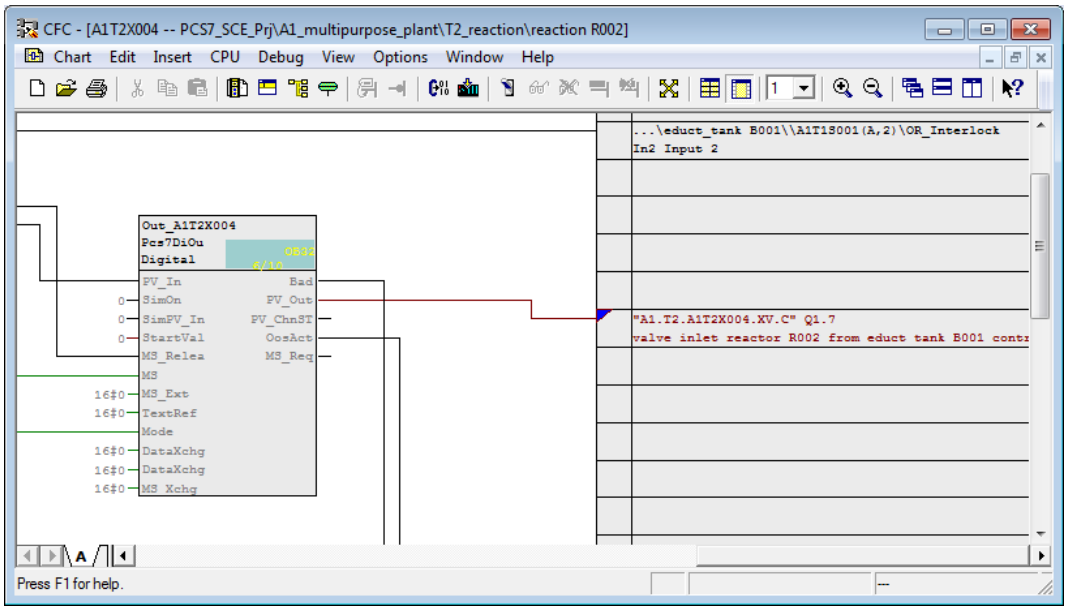


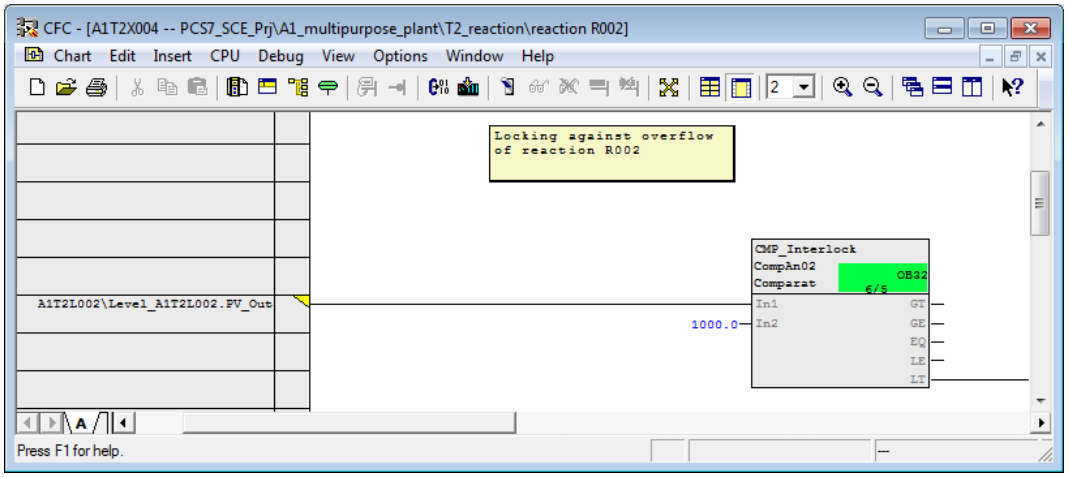
31. The newly imported CFCs are now in the hierarchy level Reactor R001. In this manner, a large number of charts can be set up quickly and effectively. The interesting aspect of this method is that the changes in the charts are not performed individually but by means of the import file in table form. Nevertheless, each individual chart can be viewed with the CFC editor afterwards.



32. Now open the newly created CFCs and check the input signals, the output signals and the block names. Textual interconnections for CFCs that already exist should be closed.

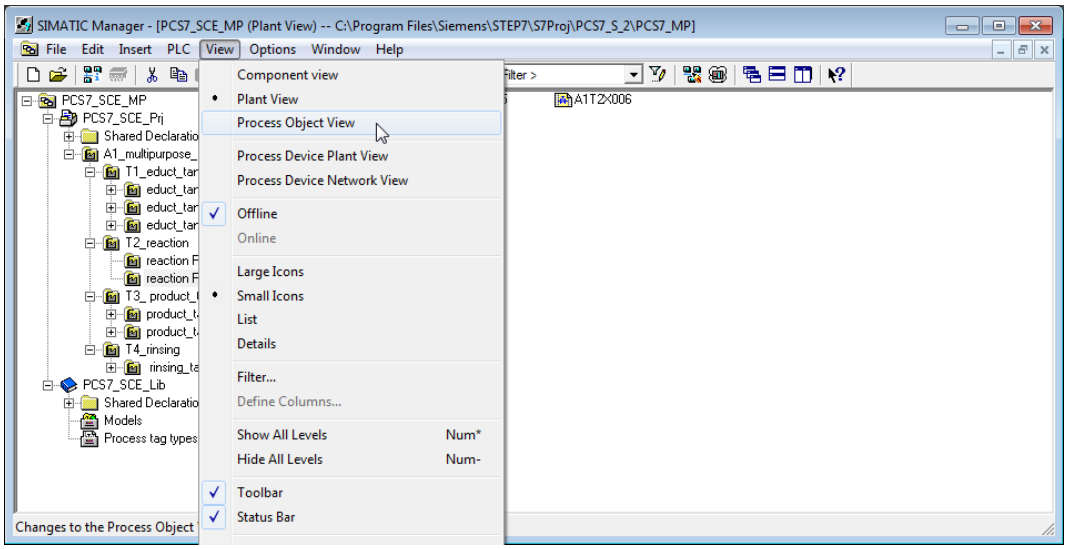




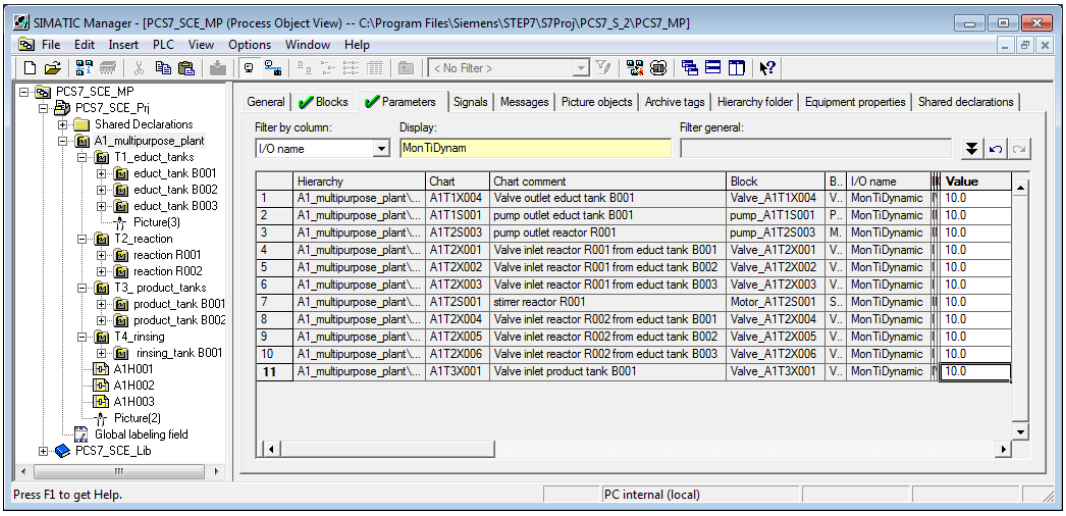


33. Another method for making changes in several charts that are already set up without opening them is the process object view.

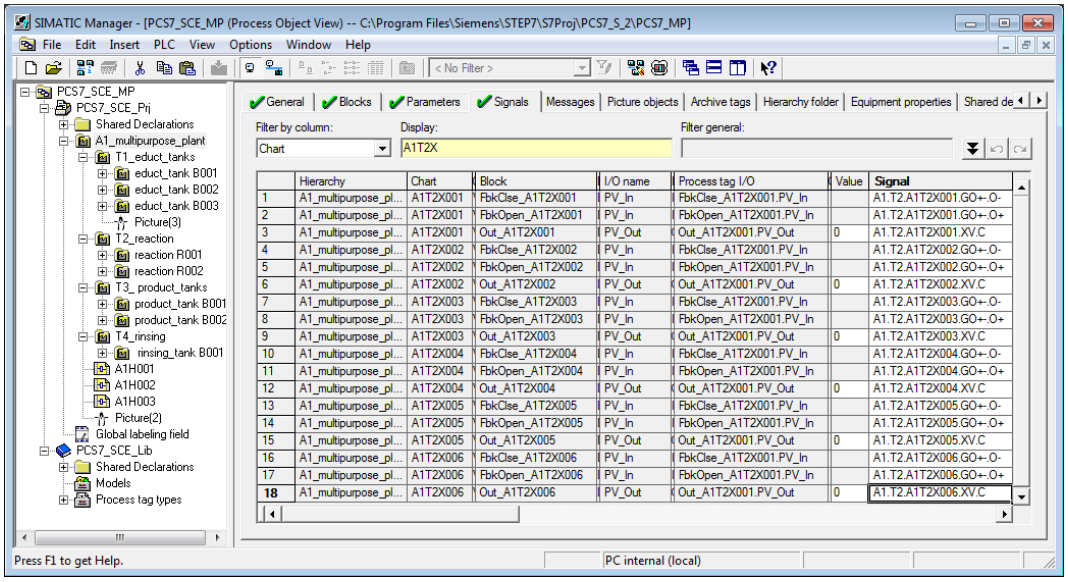
    (→ View→ Process Object View)



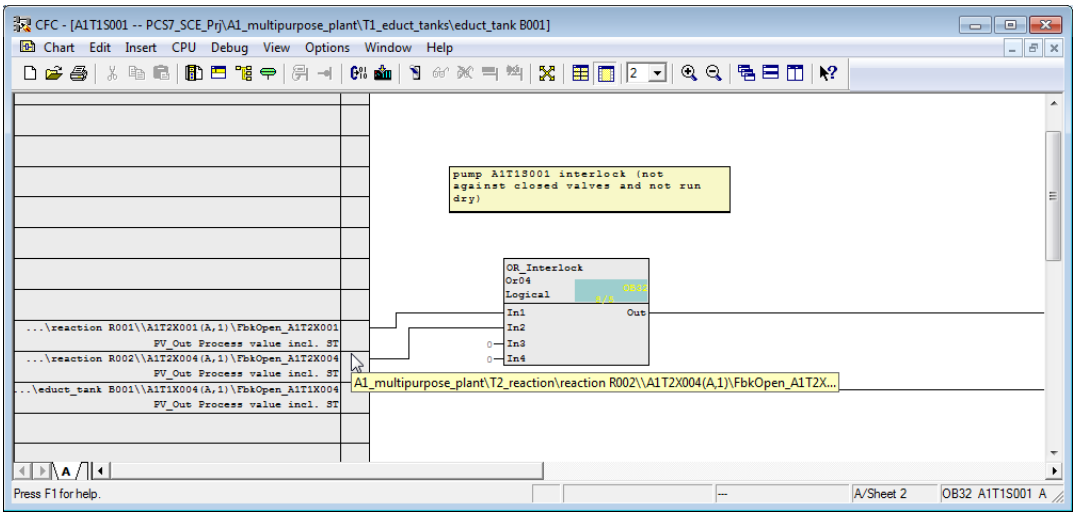
34. By setting a filter for the I/O ‘MonTiDynamic‘ in ‘Parameter‘ tab, the value of a parameter can be changed for several CFCs, for example. Only the elements are always displayed that are located below the hierarchy level selected in the left side of the window and that correspond to the filter criteria. Here, change the value for all displayed I/Os to ‘10.0‘. (→ A1_multi_purpose plant → I/O name → MonTiDynam → Value → 10.0)



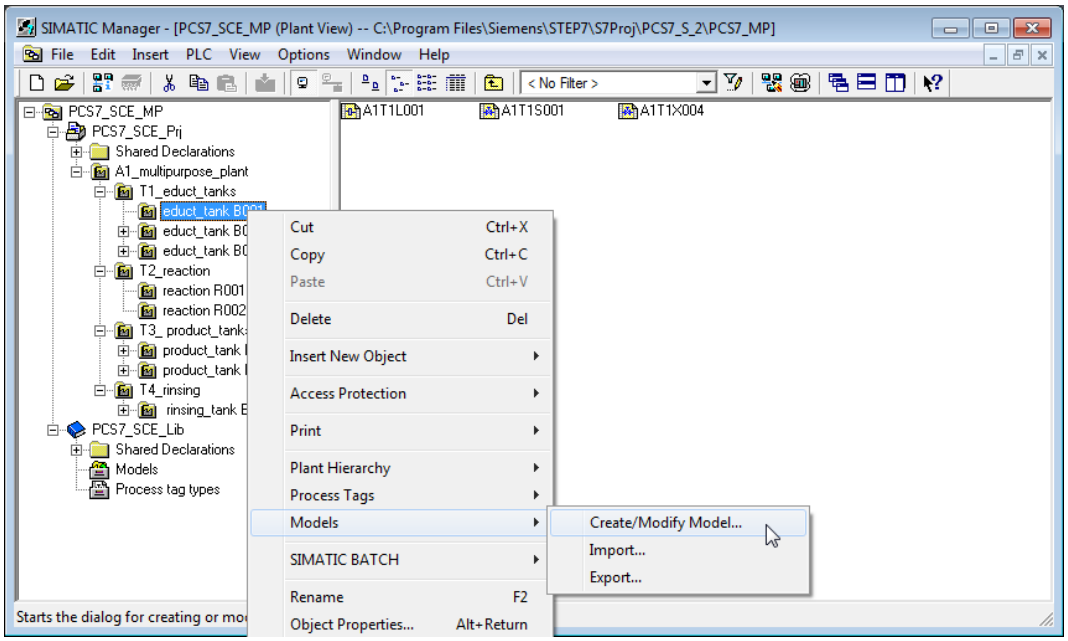
35. By utilizing the ‘Parameters‘ or ‘Signals‘ tabs, extensive changes can be made quickly on the CFCs. In this example, however, everything is to remain unchanged, and we are returning to the plant view. (→ View → Plant View).



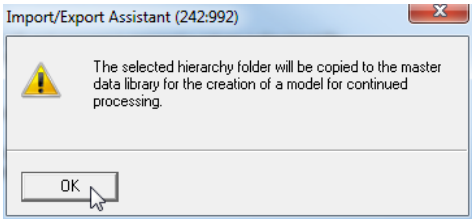
36. Before you create a model for the educt tank, complete the interlocking of the pump A1T1S001 with the valve A1T2X004 created from the process tag type (if not already done) as shown below.



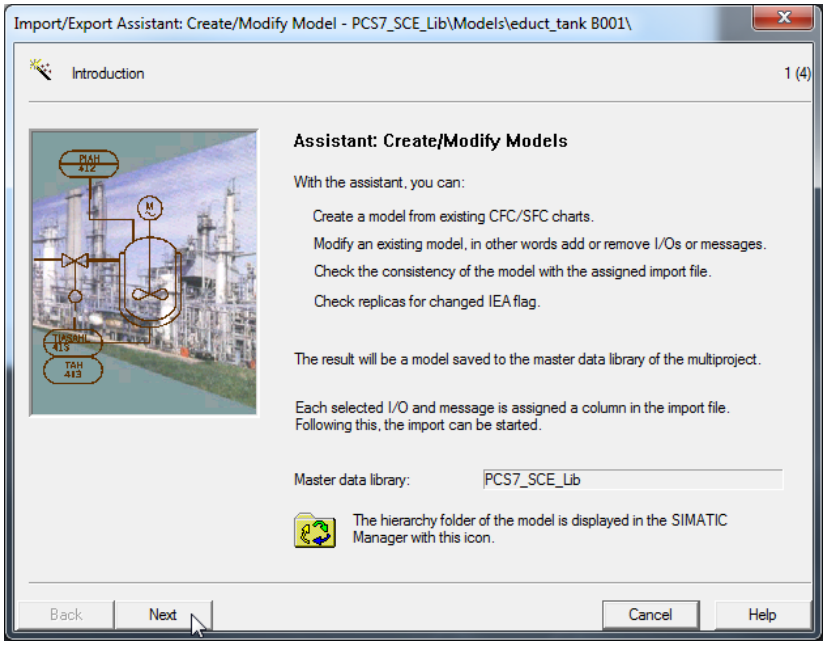
37. Educt Tank B001 with all its CFCs is used as model. First, delete figure(4) and then create a model. (→ Educt tank B001 → Models → Create/Modify Model…)



38. Confirm the message that follows with “OK“. (→ OK)

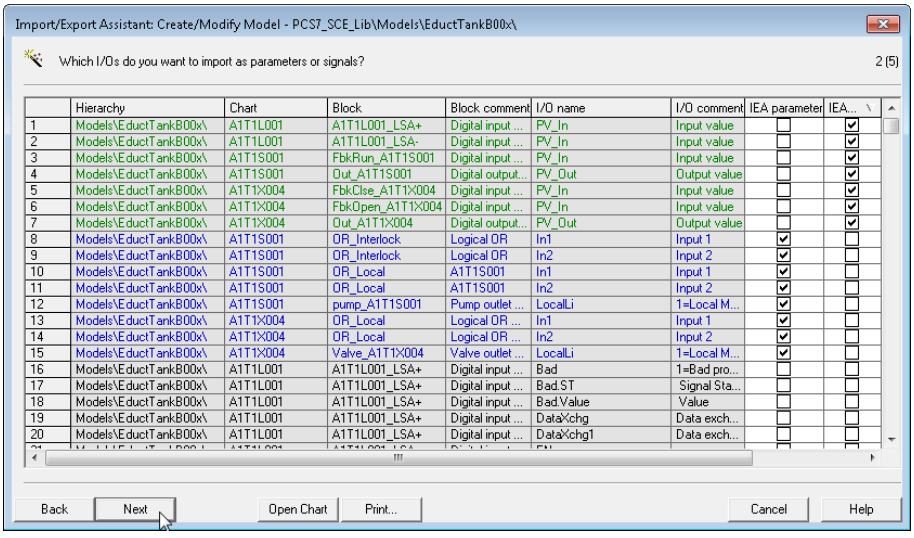


39. Confirm the introductory screen of the dialog assistant with “Next“. (→ Next)



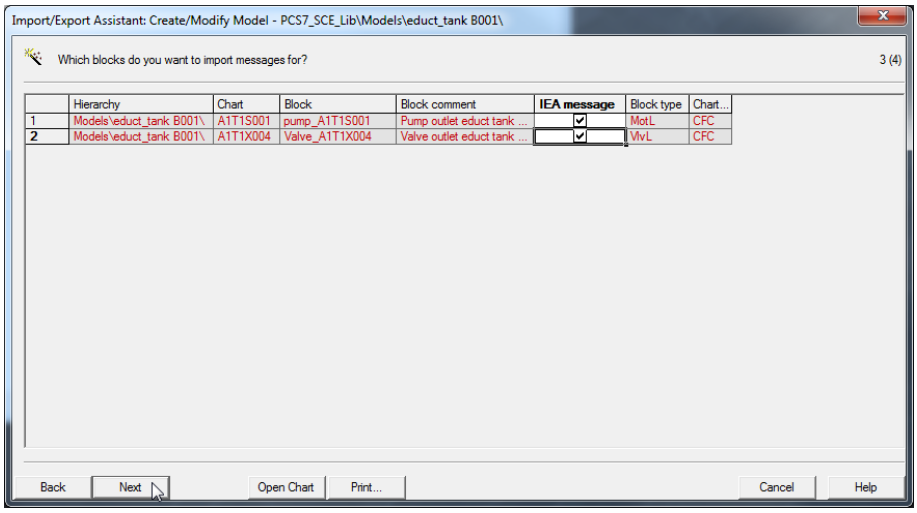
40. In the next step, specify the parameters (blue) and the signals (green) that the import/export assistant displays. Select those shown in the picture below.

(→ IEA parameter → IEA signal → Next)

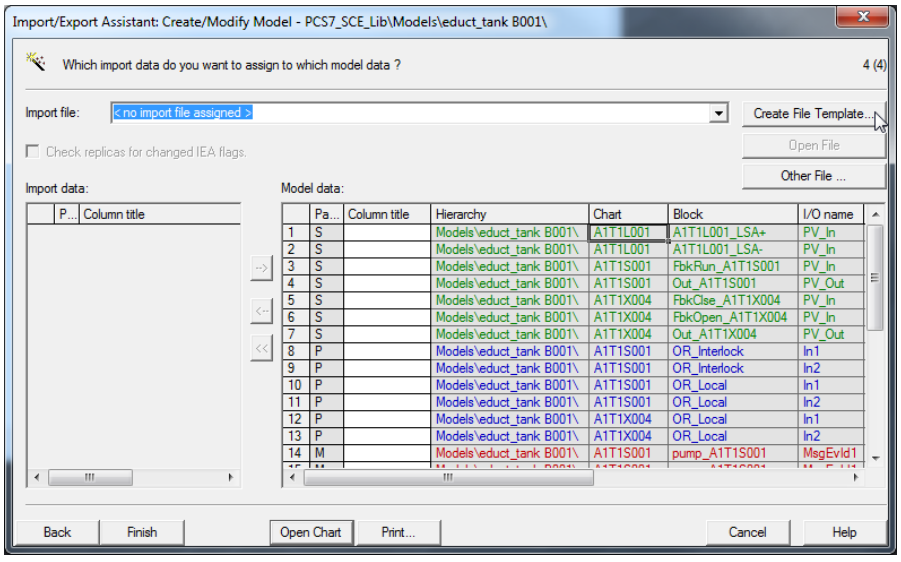


	Hierarchy	Chart	Block	Block comment	I/O name	I/O comment	IEA parameter	IEA...
1	Models\EductTankB00x\	A1T1L001	A1T1L001_LSA+	Digital input ...	PV_In	Input value	☐	☑
2	Models\EductTankB00x\	A1T1L001	A1T1L001_LSA-	Digital input ...	PV_In	Input value	☐	☑
3	Models\EductTankB00x\	A1T1S001	FbkRun_A1T1S001	Digital input ...	PV_In	Input value	☐	☑
4	Models\EductTankB00x\	A1T1S001	Out_A1T1S001	Digital output...	PV_Out	Output value	☐	☑
5	Models\EductTankB00x\	A1T1X004	FbkClse_A1T1X004	Digital input ...	PV_In	Input value	☐	☑
6	Models\EductTankB00x\	A1T1X004	FbkOpen_A1T1X004	Digital input ...	PV_In	Input value	☐	☑
7	Models\EductTankB00x\	A1T1X004	Out_A1T1X004	Digital output...	PV_Out	Output value	☐	☑
8	Models\EductTankB00x\	A1T1S001	OR_Interlock	Logical OR	In1	Input 1	☑	☐
9	Models\EductTankB00x\	A1T1S001	OR_Interlock	Logical OR	In2	Input 2	☑	☐
10	Models\EductTankB00x\	A1T1S001	OR_Local	A1T1S001	In1	Input 1	☑	☐
11	Models\EductTankB00x\	A1T1S001	OR_Local	A1T1S001	In2	Input 2	☑	☐
12	Models\EductTankB00x\	A1T1S001	pump_A1T1S001	Pump outlet ...	LocalLi	1=Local M...	☑	☐
13	Models\EductTankB00x\	A1T1X004	OR_Local	Logical OR ...	In1	Input 1	☑	☐
14	Models\EductTankB00x\	A1T1X004	OR_Local	Logical OR ...	In2	Input 2	☑	☐
15	Models\EductTankB00x\	A1T1X004	Valve_A1T1X004	Valve outlet ...	LocalLi	1=Local M...	☑	☐
16	Models\EductTankB00x\	A1T1L001	A1T1L001_LSA+	Digital input ...	Bad	1=Bad pro...	☐	☐
17	Models\EductTankB00x\	A1T1L001	A1T1L001_LSA+	Digital input ...	Bad.ST	Signal Sta...	☐	☐
18	Models\EductTankB00x\	A1T1L001	A1T1L001_LSA+	Digital input ...	Bad.Value	Value	☐	☐
19	Models\EductTankB00x\	A1T1L001	A1T1L001_LSA+	Digital input ...	DataXchg	Data exch...	☐	☐
20	Models\EductTankB00x\	A1T1L001	A1T1L001_LSA+	Digital input ...	DataXchg1	Data exch...	☐	☐

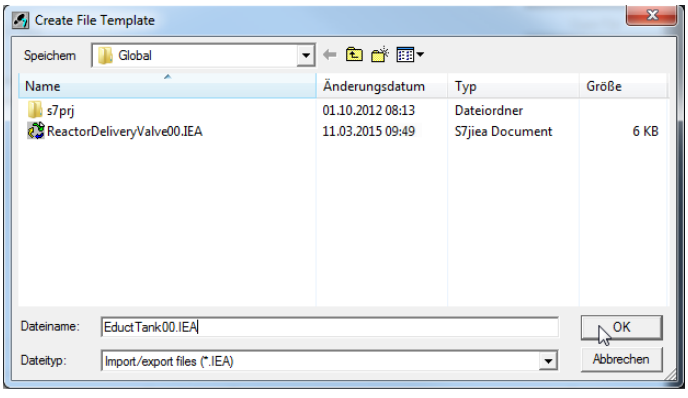
41. Next we specify the messages that are displayed in the import/export assistant.
    (→ IEA message → Next)



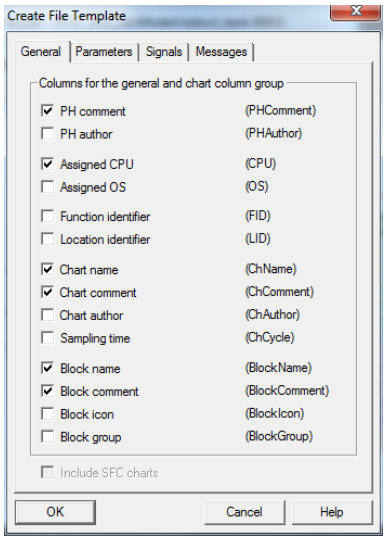
42. Now create the file template. (→ Create File Template…)

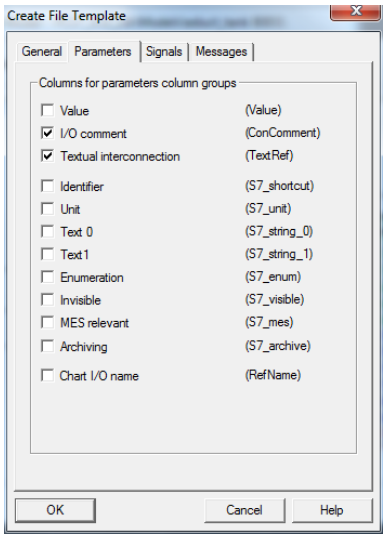


43. We are naming the file template “EductTank00.IEA“. (→ OK)

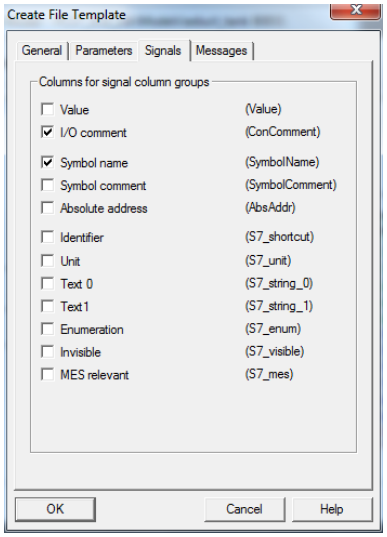


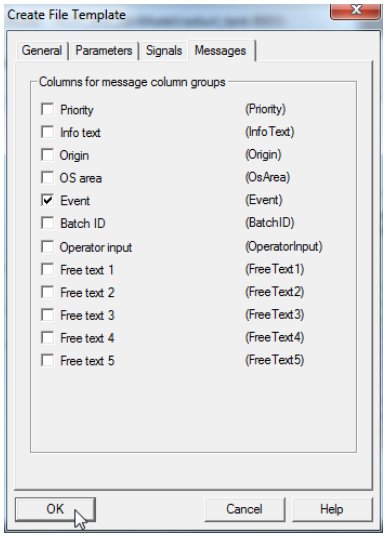
44. Next we select the columns that are displayed in general and those that are displayed for the parameters in the import file. (→ General → PH comment → Assigned CPU → Chart name → Chart comment → Block name → Block comment → Parameters → IO comment→ Textual interconnection)



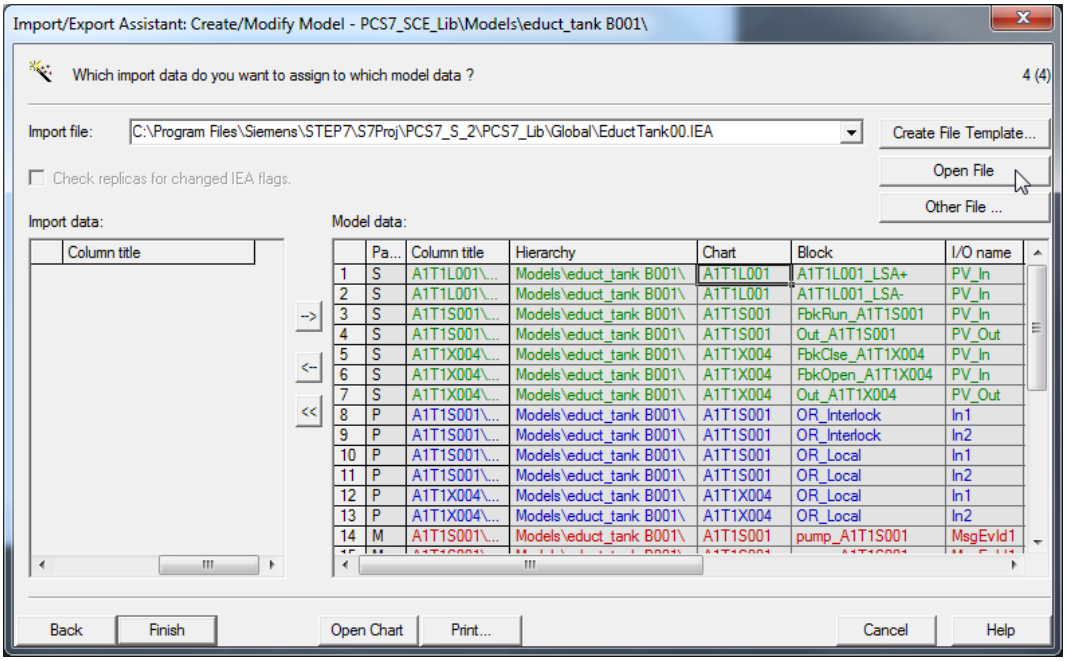


45. Here we select the columns that are displayed for the signals and the messages in the import file. (→ Signals → IO comment → Symbol name → Messages → Event → OK)



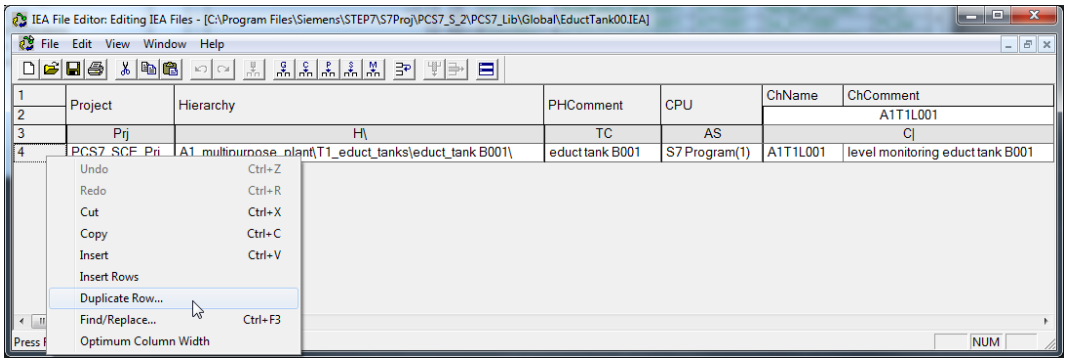


46. Now open the created file. (→ Open file)

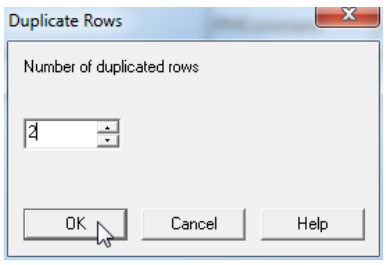


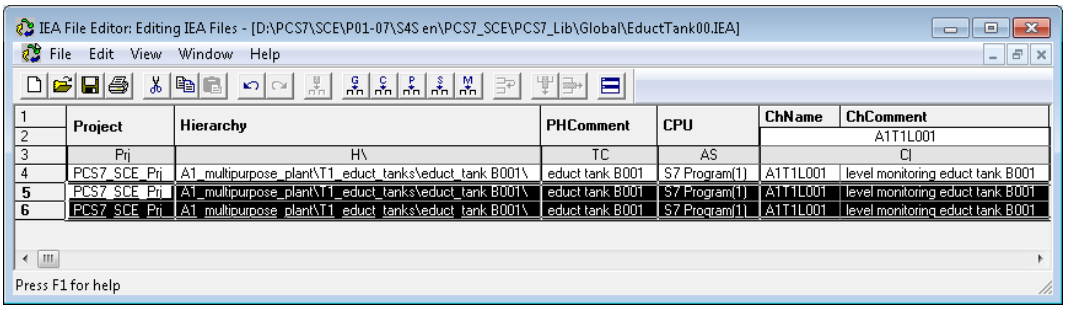
Note: As an alternative, the included import file can be utilized. Instead of Open File, select the button ‘Other file‘ and select the file that is included. With that file, the steps below can be skipped. Next step: 51.

47. The first row is again duplicated as often as models are needed. (→ Duplicate Row)

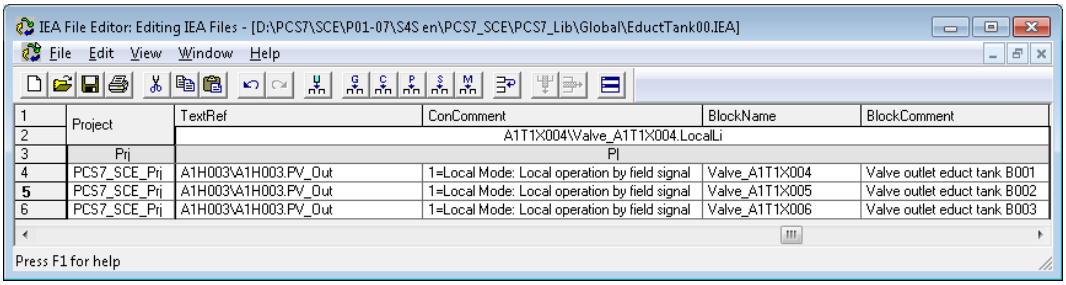


48. For Number of duplicated rows we set 2 and confirm with “OK“. (→ 2 → OK)

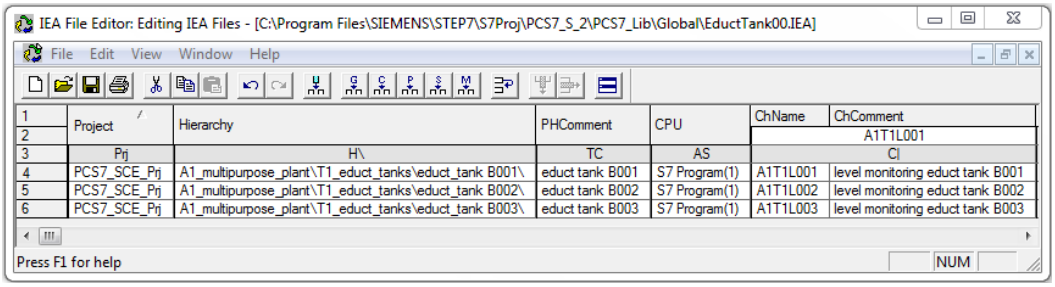


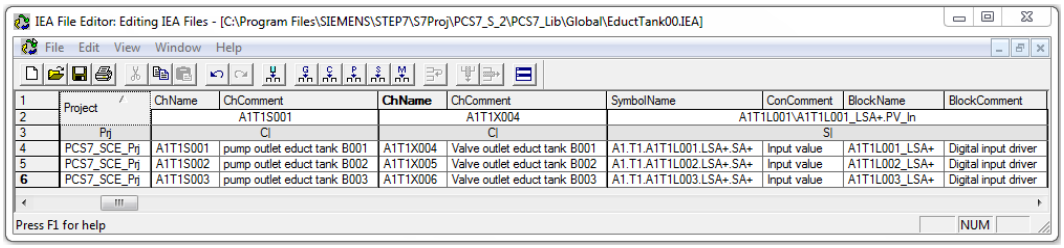


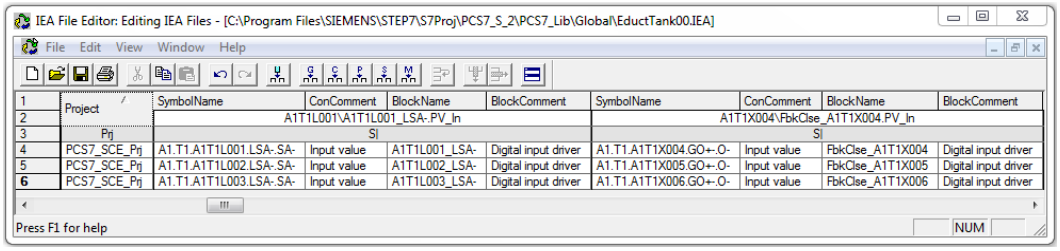
49. First, change the general information in the columns Hierarchy and PHComment. Then change the ChName and the ChComment of the CFCs. For the signals and parameters you have to adapt the SymbolName (in inverted commas for input signals and as absolute address for output signals), the BlockName/BlockComment and TextRef.

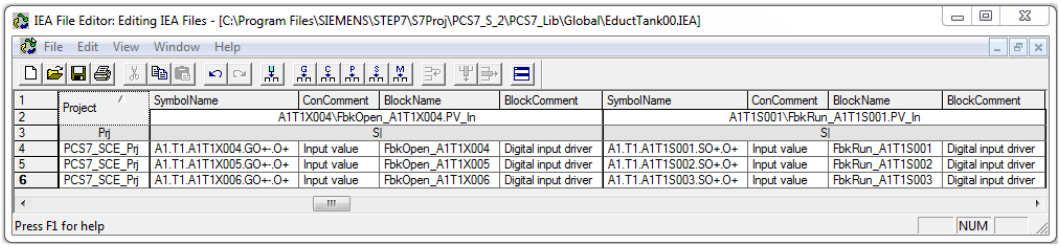


50. The messages come at the end; leave them unchanged, however. Save the file and close the editing. (→ Save →)





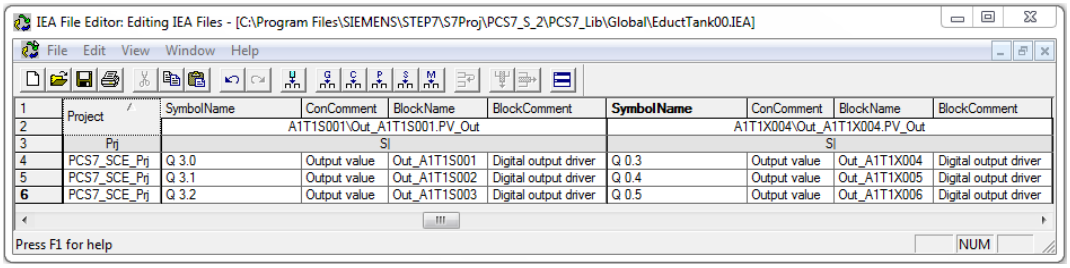



IEA File Editor: Editing IEA Files - [C:\Program Files\SIEMENS\STEP7\S7Proj\PCS7_S_2\PCS7_Lib\Global\EductTank00.IEA]

File Edit View Window Help

	Project	SymbolName	ConComment	BlockName	BlockComment	SymbolName	ConComment	BlockName	BlockComment
2		A1T1X004\FbkOpen_A1T1X004.PV_In				A1T1S001\FbkRun_A1T1S001.PV_In			
3	Prj	S|				S|			
4	PCS7_SCE_Prj	A1.T1.A1T1X004.GO+.O+	Input value	FbkOpen_A1T1X004	Digital input driver	A1.T1.A1T1S001.SO+.O+	Input value	FbkRun_A1T1S001	Digital input driver
5	PCS7_SCE_Prj	A1.T1.A1T1X005.GO+.O+	Input value	FbkOpen_A1T1X005	Digital input driver	A1.T1.A1T1S002.SO+.O+	Input value	FbkRun_A1T1S002	Digital input driver
6	PCS7_SCE_Prj	A1.T1.A1T1X006.GO+.O+	Input value	FbkOpen_A1T1X006	Digital input driver	A1.T1.A1T1S003.SO+.O+	Input value	FbkRun_A1T1S003	Digital input driver

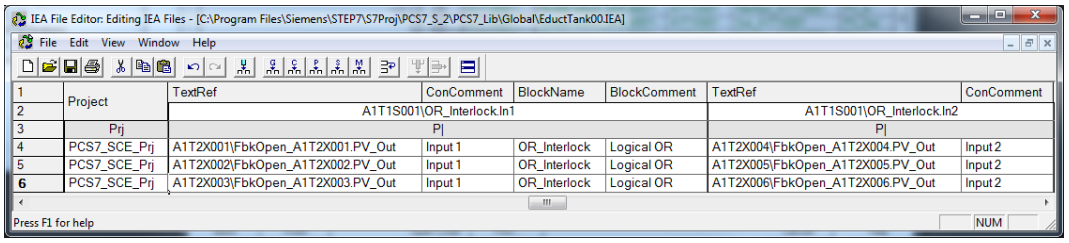
Press F1 for help NUM


IEA File Editor: Editing IEA Files - [C:\Program Files\SIEMENS\STEP7\S7Proj\PCS7_S_2\PCS7_Lib\Global\EductTank00.IEA]

File Edit View Window Help

	Project	SymbolName	ConComment	BlockName	BlockComment	SymbolName	ConComment	BlockName	BlockComment
2		A1T1S001\Out_A1T1S001.PV_Out				A1T1X004\Out_A1T1X004.PV_Out			
3	Prj	S|				S|			
4	PCS7_SCE_Prj	Q 3.0	Output value	Out_A1T1S001	Digital output driver	Q 0.3	Output value	Out_A1T1X004	Digital output driver
5	PCS7_SCE_Prj	Q 3.1	Output value	Out_A1T1S002	Digital output driver	Q 0.4	Output value	Out_A1T1X005	Digital output driver
6	PCS7_SCE_Prj	Q 3.2	Output value	Out_A1T1S003	Digital output driver	Q 0.5	Output value	Out_A1T1X006	Digital output driver

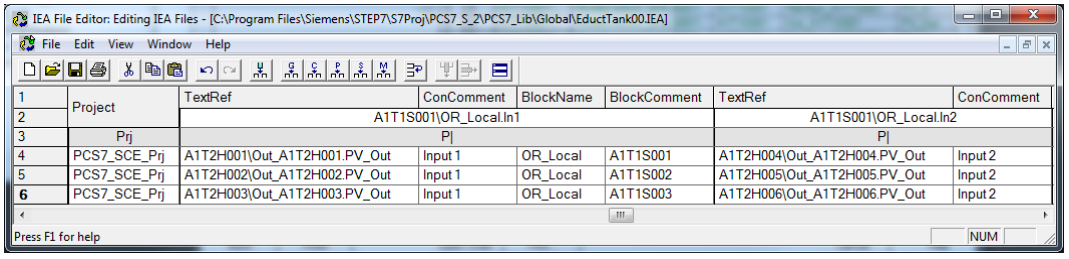
Press F1 for help NUM


IEA File Editor: Editing IEA Files - [C:\Program Files\Siemens\STEP7\S7Proj\PCS7_S_2\PCS7_Lib\Global\EductTank00.IEA]

File Edit View Window Help

	Project	TextRef	ConComment	BlockName	BlockComment	TextRef	ConComment
2		A1T1S001\OR_Interlock.In1				A1T1S001\OR_Interlock.In2	
3	Prj	P|				P|	
4	PCS7_SCE_Prj	A1T2X001\FbkOpen_A1T2X001.PV_Out	Input 1	OR_Interlock	Logical OR	A1T2X004\FbkOpen_A1T2X004.PV_Out	Input 2
5	PCS7_SCE_Prj	A1T2X002\FbkOpen_A1T2X002.PV_Out	Input 1	OR_Interlock	Logical OR	A1T2X005\FbkOpen_A1T2X005.PV_Out	Input 2
6	PCS7_SCE_Prj	A1T2X003\FbkOpen_A1T2X003.PV_Out	Input 1	OR_Interlock	Logical OR	A1T2X006\FbkOpen_A1T2X006.PV_Out	Input 2

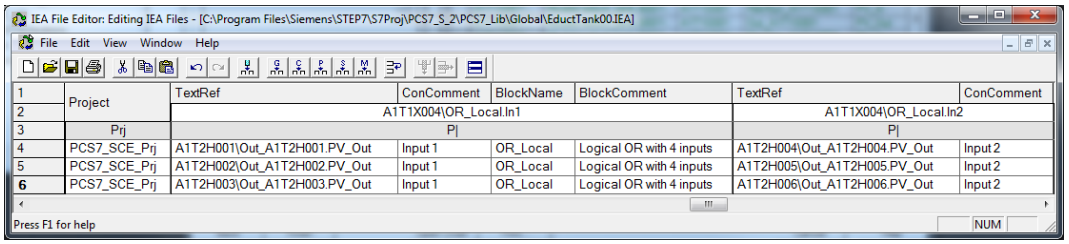
Press F1 for help NUM


IEA File Editor: Editing IEA Files - [C:\Program Files\Siemens\STEP7\S7Proj\PCS7_S_2\PCS7_Lib\Global\EductTank00.IEA]

File Edit View Window Help

	Project	TextRef	ConComment	BlockName	BlockComment	TextRef	ConComment
2		A1T1S001\OR_Local.In1				A1T1S001\OR_Local.In2	
3	Prj	P|				P|	
4	PCS7_SCE_Prj	A1T2H001\Out_A1T2H001.PV_Out	Input 1	OR_Local	A1T1S001	A1T2H004\Out_A1T2H004.PV_Out	Input 2
5	PCS7_SCE_Prj	A1T2H002\Out_A1T2H002.PV_Out	Input 1	OR_Local	A1T1S002	A1T2H005\Out_A1T2H005.PV_Out	Input 2
6	PCS7_SCE_Prj	A1T2H003\Out_A1T2H003.PV_Out	Input 1	OR_Local	A1T1S003	A1T2H006\Out_A1T2H006.PV_Out	Input 2

Press F1 for help NUM

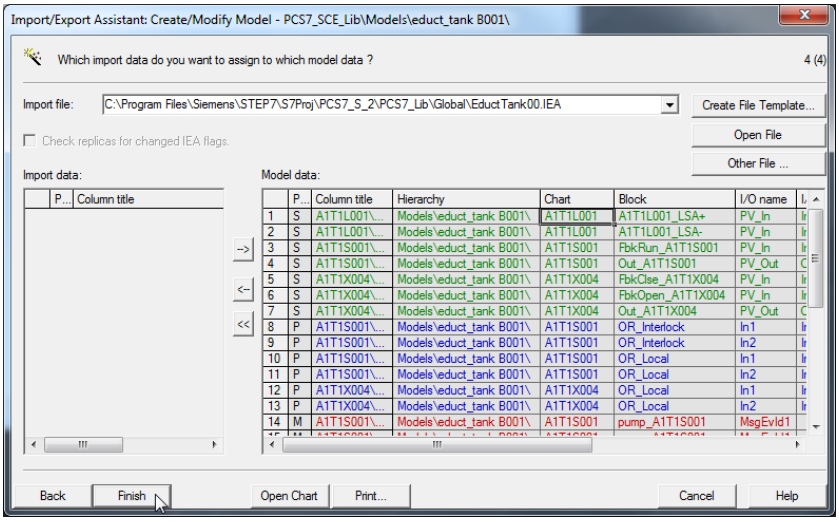

IEA File Editor: Editing IEA Files - [C:\Program Files\Siemens\STEP7\S7Proj\PCS7_S_2\PCS7_Lib\Global\EductTank00.IEA]

File Edit View Window Help

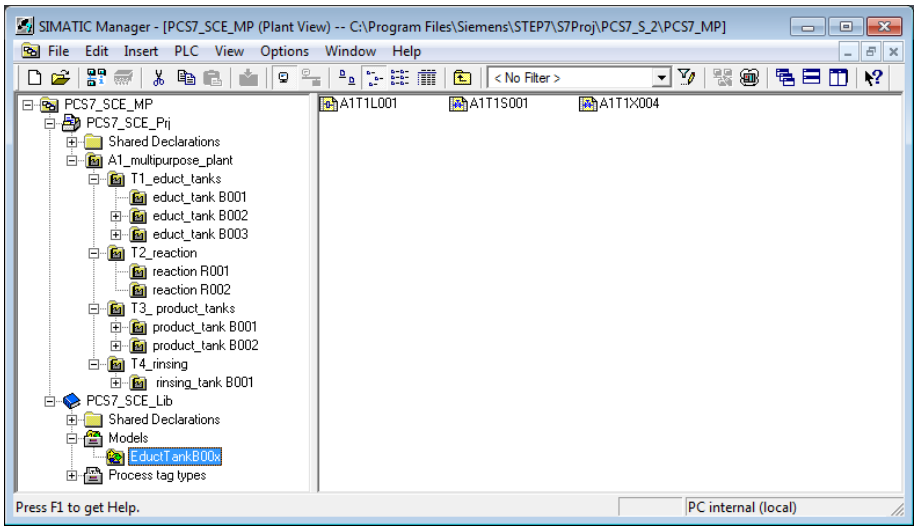
	Project	TextRef	ConComment	BlockName	BlockComment	TextRef	ConComment
2		A1T1X004\OR_Local.In1				A1T1X004\OR_Local.In2	
3	Prj	P|				P|	
4	PCS7_SCE_Prj	A1T2H001\Out_A1T2H001.PV_Out	Input 1	OR_Local	Logical OR with 4 inputs	A1T2H004\Out_A1T2H004.PV_Out	Input 2
5	PCS7_SCE_Prj	A1T2H002\Out_A1T2H002.PV_Out	Input 1	OR_Local	Logical OR with 4 inputs	A1T2H005\Out_A1T2H005.PV_Out	Input 2
6	PCS7_SCE_Prj	A1T2H003\Out_A1T2H003.PV_Out	Input 1	OR_Local	Logical OR with 4 inputs	A1T2H006\Out_A1T2H006.PV_Out	Input 2

Press F1 for help NUM

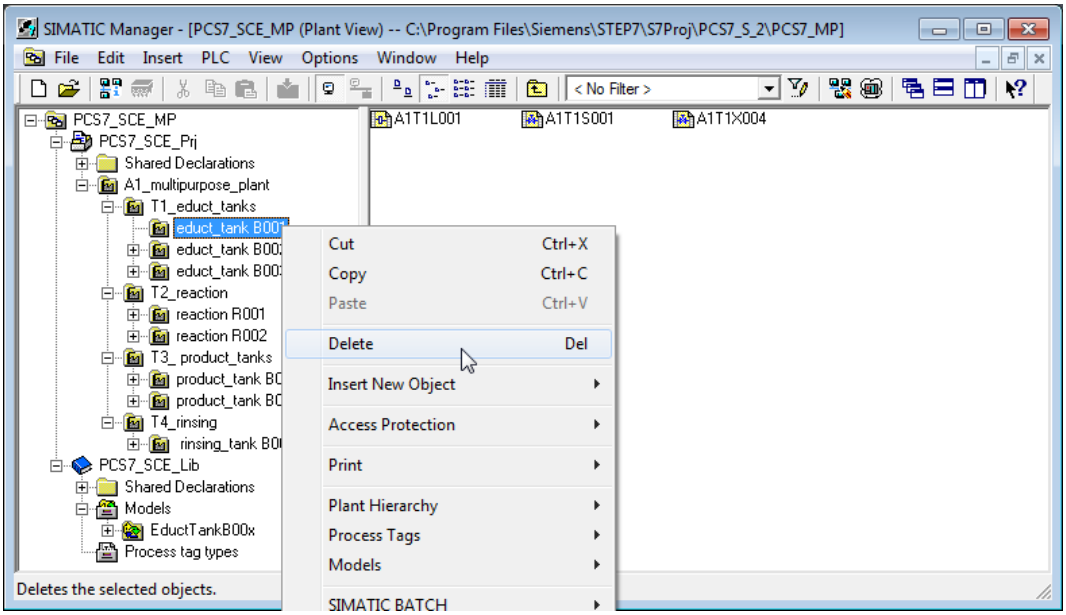
51. The Assistant is exited with “Finish“. (→ Finish)



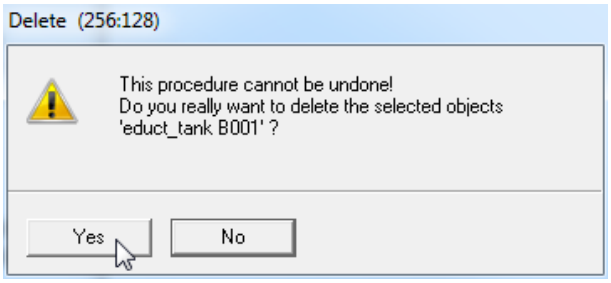
52. The newly created model is located in the project library in the folder Models. Here, the model that was created is renamed to “EductTankB00x”.



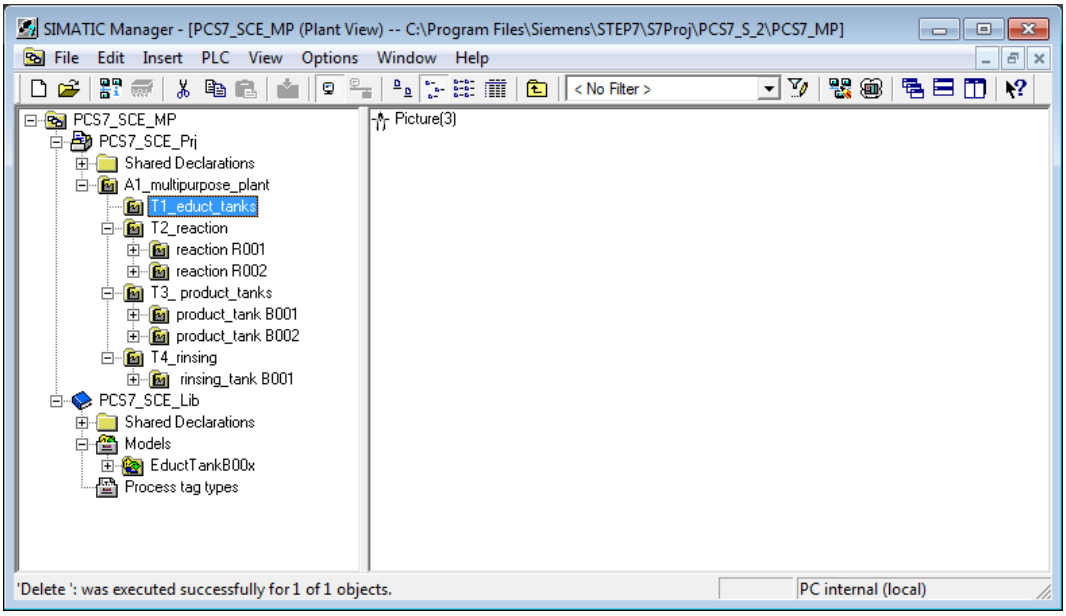
53. Before starting the import, the hierarchy folders B001 to B003 including the CFCs they contain have to be deleted. (→ educt_tank B00x → Delete)



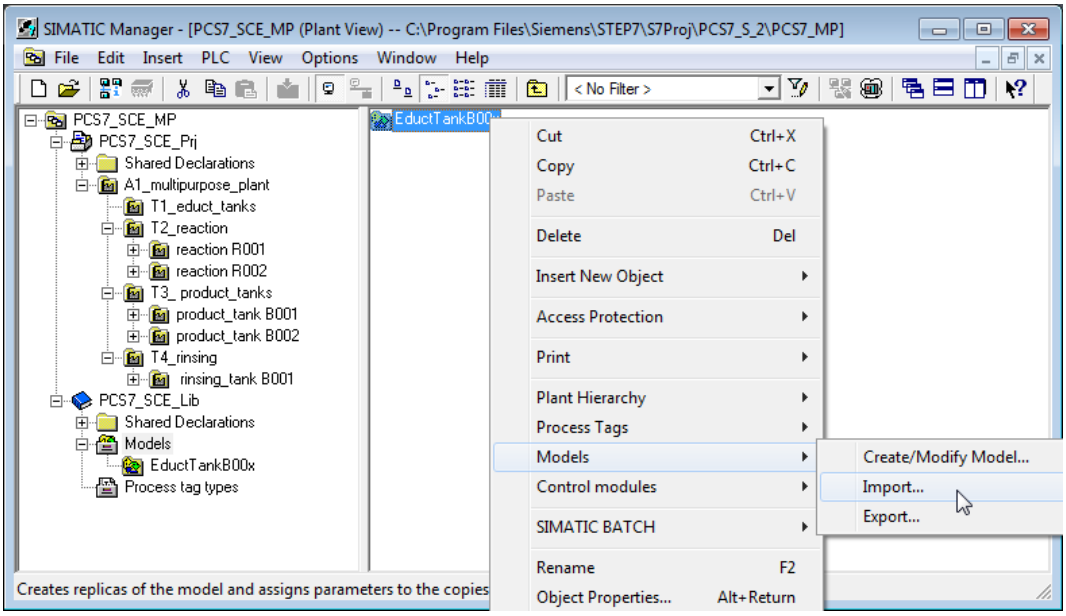
54. Confirm the warning with “Yes“. (→ Yes)



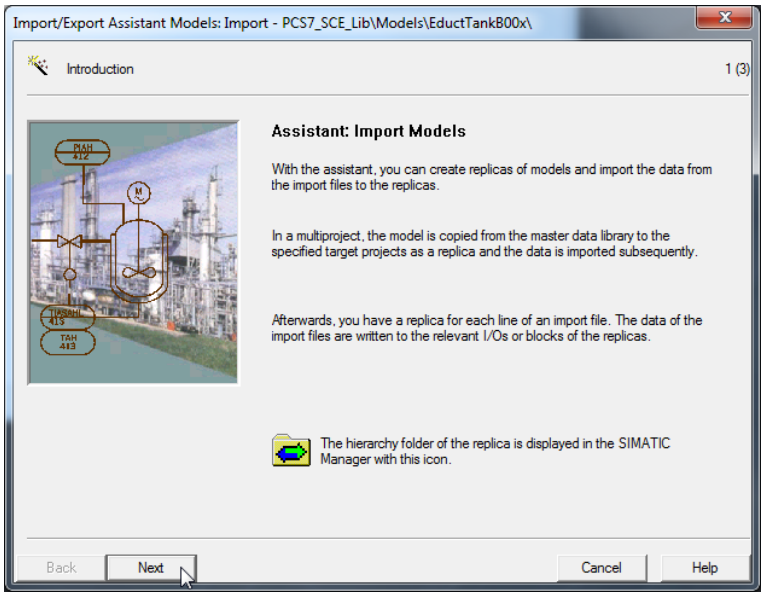
55. After the deletion, the plant hierarchy looks like this.



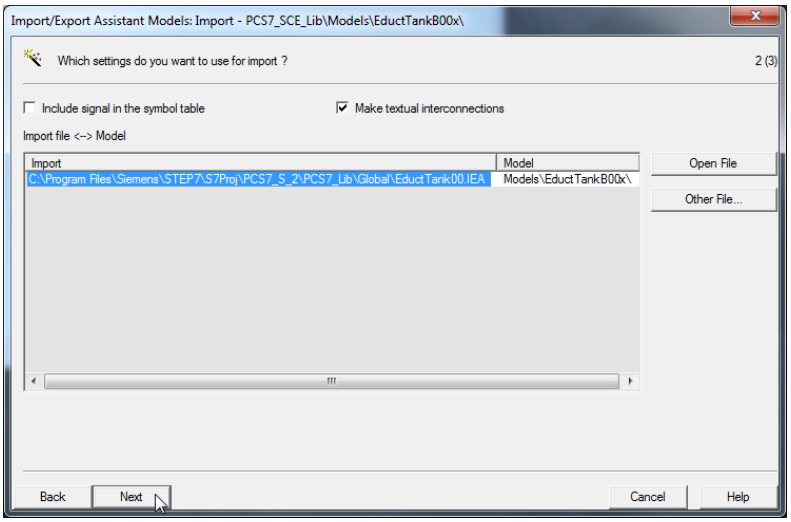
56. Now we can start importing the model. (→ EductTankB00x → Models → Import…)



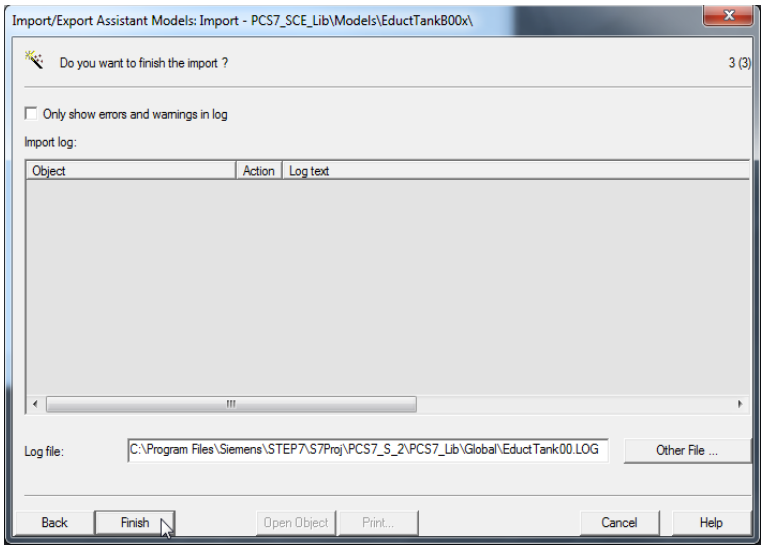
57. Confirm the start screen of the import/export assistant with “Next“. (→ Next)



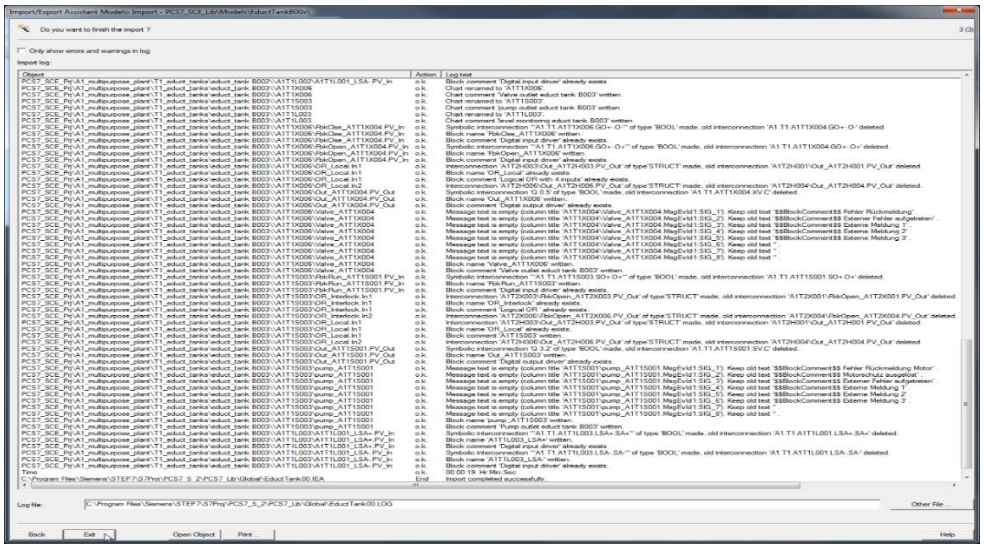
58. Check “Make textual interconnections“ and click on “Next“. (→ Make textual interconnections → Next)



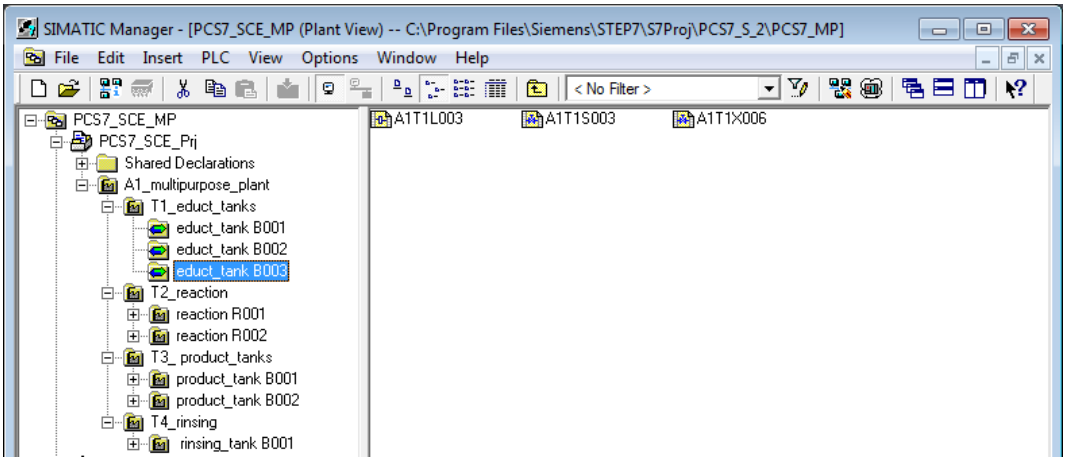
59. The assistant is now finished and the import is started. (→ Finish)



60. The import process is logged and the result is displayed.(→ Exit)



61. The imported models are now present in the plant hierarchy.



62. Check to see if the textual interconnections with the existing CFCs are closed.

Table 1: Textual interconnections in chart ‘A1T1S001’

Input:	**Textual interconnection:**	**Inverted**
MotL.Pumpe_A1T1S001 .LocalLi	A1H003\A1H003.PV_Out	No
Intlk02.Permit.In01	A1H001\A1H001.PV_Out	No
Intlk02.Protect.In01	A1H002\A1H002.PV_Out	No
Or04.Or_Interlock.In1	A1T2X001\FbkOpen_A1T2X001.PV_Out	No
Or04.Or_Interlock.In2	A1T2X004\FbkOpen_A1T2X004.PV_Out	No
Or04.Or_Local.In1	A1T2H001\Out_A1T2H001.PV_Out	No

Table 2: Textual interconnections in chart ‘A1T1X004’

Input:	**Textual interconnections:**	**Inverted**
VlvL.Pumpe_A1T1X004. LocalLi	A1H003\A1H003.PV_Out	No
Intlk02.Permit.In01	A1H001\A1H001.PV_Out	No
Intlk02.Protect.In01	A1H002\A1H002.PV_Out	No
Or04.Or_Local.In1	A1T2H001\Out_A1T2H001.PV_Out	No

Table 3: Textual interconnections in chart ‘A1T2H001’

Input:	**Textual interconnection:**	**Inverted**
Or08.Or_A1T2H001.In7	A1T1L001\A1T1L001_LSA-.PV_Out	Yes

EXERCISES

In the exercises we apply what we learned in the Theory section and in the Step by Step Instructions. The existing multi-project from the step by step instructions (PCS7_SCE_0107_R1505_en.zip) will be utilized and expanded.

The tasks in this exercise supplement the plant with all objects not implemented so far. It is up to you where you want to utilize the tools for importing plant design data. Effective utilization of importing plant design data does not only depend on the plant structure, but also on the mapping of this structure in the plant hierarchy. With some practice, you will improve your knowledge regarding meaningful plant designations and the structure of the plant hierarchy.

TASKS

1. Complete the following CFCs in Reactor R001:
   - A1T2H002 and A1T2H003
   - A1T2H013 and A1T2H015
   - A1T2X007.
2. Check open textual interconnections between the manual controls in the reactor and other CFCs in Reactor R001. To this end, you can also utilize the function ‘Close textual interconnections’ under Options in the CFC Editor. In the result, the interconnections that could not yet be closed are displayed. With a double click or by pressing the button “Go to“, select an interconnection that is still open and correct it manually.

   Note: Not all open textual interconnections can be closed here. Most important are the connections within Reactor R001.
3. Now, create a model of Reactor R001. Delete the folder Reactor R002 and import the model. Reactor R001 is omitted automatically because the folder already exists. Should you delete it, it will also be generated from the model.
4. Next, create a model of Product Tank B001. Delete at least the folder Product Tank B002 and import the model.
5. Now set up the missing CFCs for the rinse tank:
   - A1T4L001
   - A1T4S001
   - A1T4X001, A1T4X002, A1T4X003 and A1T4X004.
6. Interconnect the manual control Rinse in a way that the rinsing water flows from the rinse tank into the reactor and right away back into the rinse tank.
7. Check whether textual interconnections are still open.
8. Finally, check all CFCs for correct designations and correct connections. For the first task, it is best to utilize the process object view. Always select one CFC in the left window while checking the name of the blocks in the ‘Blocks’ tab in the right window. To look for errors, however, you should use the simulation.